Abstract

This document introduces the concept of Virtual Customer Premises Equipment (VCPE). Such concept was first proposed in Broadband Forum (BBF) as Network Enhanced Residential Gateway (NERG). The concept is further expanded as not only referring to virtual CPE of residential network, but all the virtual network and service functions shifted from the customer side to the operator side. Deployment of VCPE in some typical DMM (Distributed Mobility Management) scenarios brings specific requirements and even protocol extension in DMM. In this document, we will first explain the motivation and advantages of VCPE. A usecases of VCPE in the community Wi-Fi deployment is further discussed so as to explain the deployment of VCPE in a DMM scenario. Three models of field deployment of VCPE are discussed afterwards to indicate the possible CP/DP decomposition requirement and protocol extension.

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 April 21, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.
This document introduces the concept of VCPE. The concept of VCPE is to shift most of the networking and service functionalities from the customer side to the network side. In this way, the customer side’s equipment, that is the pCPE (Physical Customer Premises Equipment), can be simplified. The VCPE refers to one or a set of equipments at the network side to execute the networking and service functionalities used to be executed at the CPE. In such architecture, the CPE can be a simple L2 switch, which is only responsible for forwarding packets to a certain next hop. The concept of VCPE was first introduced in BBF as NERG (WT-317), which mainly focuses on shifting some of the functionalities of a residential gateway to the operator’s network, for enabling network based features. The aim is to facilitate the deployment, maintenance and evolution of both existing and new capabilities without adding complexity to the RG and/or the home network.

Figure 1 shows the architecture of the pCPE and the VCPE.
In this document, we would like to further propose such concept in the following aspects:

(1) Motivation and advantages of VCPE.

(2) Usecases of VCPE. A usecase of VCPE in the community Wi-Fi is explained in detail.

(3) Models of VCPE deployment. We propose three models for the field deployment of VCPE.

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. Motivation and Advantage of VCPE

The motivation and advantage of introducing VCPE can be concluded as follows:

(1) It will greatly speed up the service launching period. Since most of the complicated functions are located at the VCPE in the network side, operators have more power over services. Benefitting from the recent NFV (Network Function Virtualization) and cloud technologies, VCPE can be accomplished using SFC in the virtual network, where different services can act as different VNFs (Virtual Network Functions). Operators only need to add new VNFs on the VCPE side to launch new services to the customers. In this way, Operators can provide a variety of services through the network.

(2) It will reduce the cost of the pCPE. By shifting most of the complicated functions from the customer’s side to the operator’s side, the cost of the pCPE can be reduced significantly. Such reduction can be remarkable in the enterprise network, since network functions, such as Firewall and NAT (Network Address Translator) at the customer side can be expensive. In the meantime, the cost of
upgrading tens of thousands of pCPE when launching new services can be saved, since only software upgrade at the VCPE side is required.

(3) It will simplify the maintenance of the pCPE. Since most of the complicated functionalities are shifted to the network side, the maintenance of the pCPE can be greatly simplified. On-line maintenance is possible in lots of cases since the pCPE is only a L2 devices and can be considered transparent to the operators.

(4) It will provide user-define-network experience. By introducing SFC concept into the VCPE, users can define his own service order and sequence. Therefore, customers can enjoy the self-defined services over the public network.

4. Use case of VCPE

The concept of VCPE can be used in multiple scenarios. In this section, we will propose a usecase of VCPE when deploying community Wi-Fi.

The community Wi-Fi is a new service that operators provide to leverage unused capacity on existing residential Wi-Fi infrastructure to offer Wi-Fi network access to visitors and passers by near the neighbourhood. An operator can also use this excess capacity to offer services to retail and roaming-parter operators’ subscribers. The residential subscribers accessing the network from inside their homes have prioritized access to the Wi-Fi resources. The residential Wi-Fi infrastructure is configured in a manner that allows for a secure and independent access channel to retain service quality, safety, and privacy for both residential and visitor customers. Roaming users are only allowed to use the Wi-Fi network capacity that is not currently used by the subscriber at home.

Basically, the wireless Access Point (AP) in the home will provide two networks: a private one for the home owner/subscriber, and a community network for on-the-go subscribers passing through the neighborhood. Home users can have all of their Wi-Fi devices (smartphone, tablet, etc.) automatically connect to the private network. In the meantime users travel outside can connect to the community network, and can roaming through different APs supporting community Wi-Fi as he/she is moving. The community Wi-Fi service is a typical usecase of DMM.

Deploying community Wi-Fi on the pCPE means upgrading tens of thousands of existing pCPE devices at the customer side, which is not cost-effective and may bring extra complexity for maintainance. Therefore VCPE becomes an optimized solution for such deployment. In such deployment, the private users access to the pCPE (which is the
The public users are roaming through different pCPEs. The traffic all goes through the tunnel from the pCPE to the VCPE. The deployment of VCPE in the community Wi-Fi scenario brings specific requirement and protocol extensions to DMM. The deployment model of VCPE and its possible influence to DMM is further discussed in the following section.

5. Models of VCPE Deployment

There are multiple models when deploying VCPE in use cases as are discussed in the previous section. In this document, we conclude the deployment of VCPE into three models. In the first model, a logical instance of VCPE is deployed in the cloud for each pCPE instance. That is, the pCPE and VCPE is deployed in an 1:1 manner. All traffic from pCPE goes through the vCPE.

![Figure 2: VCPE deployment model NO.1: Logical Instance of VCPE](image)

In the second model, vCPE is modeled service function chains in Gi-LAN. BNG knows how to classify the traffic from a given CPE with the help of the control plane, and run it through the service chain. In such model, the CP/DP interface should be used between the control plane (which might be the controller) and the pCPE.
Figure 3: VCPE deployment model NO.2: VCPE as SFC

The third model is almost the same with the second one, except that the BNG is also CP/DP decomposed. In this model, The control plane is composed of the controller of the pCPE and the control plane of the BNG. The CP/DP interface is used between the controller and the pCPE, and between the control plane and the data plane of the BNG. Both of model No.2 and No.3 may have specific requirement and protocol extensions for the CP/DP interface due to the usecase of VCPE.
SDN (Software Define Network) controllers can also be introduced in the third model. In which case, all of the pCPEs and the BNG data plane (BNG DP) can be controlled by the SDN-controller. When the customer selects a set of services, the SDN-controller will inform the pCPE and the BNG DP to direct the traffic flow to a certain SFC.
6. VCPE Deployment for Community Wi-Fi

In this section, we will discuss about the VCPE deployment for Community Wi-Fi in detail. In the following deployment, we assume the VCPE is deployed following the third model we discussed in section 5. That is, the VCPE is a bunch of SFCs at the operator side behind the BNG. The pCPEs and BNG-DP are all controlled by a mutual control plane. The FPC protocol is used between the control plane and the pCPEs, and that and the BNG-DP.

As we discussed in section 4, Community Wi-Fi can be deployed with the help of deploying VCPE. In order to provide the Community Wi-Fi service, the pCPE should provide two SSIDs, one for the public Wi-Fi users, and the other for the private Wi-Fi users. Packets from different SSID are marked with different VLAN ID. The VCPE should know of the corresponding relation between the SSID and the VLAN ID, so as to provide distinguished services to the publice users and the private users. For instance, the private users should experience a better QoS than the publice ones. In the meantime, the private users and the public users may choose different SFC in the VCPE. All of these different services are classified based on the VLAN ID.

Such deployment requires the FPC client to support the following task:
1) The FPC client should be able to set specific VLAN to each SSID.
2) The FPC client should be able to set the QoS for specific VLAN ID.
3) The FPC client should be able to inform the agent the specific SFC for each VLAN ID.
4) The FPC client should be capable of instruct the agent to handle the MN hand-over of the public Wi-Fi users.

In the meantime, such deployment requires the FPC agent to support the following task:

1) The FPC agent should be able to set specific VLAN to each SSID following the command from the client.
2) The FPC agent should be able to set the QoS for specific VLAN ID following the command from the client.
3) The FPC agent should be able to direct the traffic for specific VLAN ID to a certain SFC following the command of the client.
4) The FPC agent should be able to handle the MN hand-over of the public Wi-Fi users.

7. 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 simplify the maintenance 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.

8. Informative References

Authors’ Addresses

Qiao Fu (editor)
China Mobile
Xuanwumenxi Ave. No.32
Beijing
China

Email: fuqiaol@outlook.com

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

Email: denghui@chinamobile.com
Abstract

The DMM working group is currently looking at solutions for mobility management based on control and data plane separation. In this context, the network entities in charge for the mobility control function are separated from the data plane nodes, responsible mainly for, but not limited to, data packet forwarding. This vision allows for the design of different interaction methods between the mobility control function and the data plane nodes. One of such mechanisms being currently discussed devises an SDN-based abstraction layer between control and data planes, embodied by an SDN Network Controller. This latter interfaces to the mobility control function through a north-bound interface (NBI), and programs the underneath infrastructure through a south-bound interface (SBI), e.g., OpenFlow.

This draft describes multiple deployment scenarios for NBI and SBI interworking, given that multiple network controllers could be deployed in order to meet performance requirements e.g., related to latency.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

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."
time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 7, 2016.

Copyright Notice

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

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

Table of Contents

1. Introduction .................................................. 2
2. Terminology ................................................... 5
3. Control plane and data plane separation: deployment scenarios 6
   3.1. Scenario 1 ............................................... 6
   3.2. Scenario 2 ............................................... 7
   3.3. Scenario 3 ............................................... 7
4. References ..................................................... 8
   4.1. Normative References ..................................... 8
   4.2. Informative References ................................. 8
Authors’ Addresses ............................................... 8

1. Introduction

The Distributed Mobility Management (DMM) working group is currently investigating solutions, which are based on the separation of mobility management functions’ Control- and Data-Plane. The network functions in charge of the mobility control are separated from the data plane nodes, responsible mainly for, but not limited to, data packet forwarding. In this scenario, it is possible to design different mechanisms for the interaction between the mobility control function and the data plane nodes.

One of these mechanisms devises a Software Defined Networking (SDN) abstraction layer between control and data planes, embodied by an SDN Network Controller (NC). The NC enforces instructions from different mobility control functions, which operate on top of a Network
Controller and hold the mobility states, to Data-Plane nodes (DPNs), such as switches or routers, within a defined network scope. DPNs in different networks may be controlled by physically separated NCs. The NC configures Data-Plane nodes with forwarding and policy enforcement related rules, either with the accuracy of a single data flow or as aggregated rule, such as for an IP address prefix.

An example of a Control-Plane function is the Control-Plane of a Local Mobility Anchor (LMA-C) as per the Proxy Mobile IPv6 (PMIPv6) architecture [RFC5213] or a Packet Data Network (PDN) Gateway’s Control-Plane functions as per the 3rd Generation Partnership Project’s architecture for cellular mobile communications. These functions perform protocol operation with remote Control-Plane functions for mobility management, e.g. the Control-Plane of a Mobile Access Gateway (MAG) as per the PMIPv6 protocol.

Control-Plane functions associated with mobility management need to interact with the NC to enforce forwarding and policy rules in the Data-Plane. By taking again the PMIPv6 architecture as an example, the Data-Plane nodes have the role of the Data-Plane functions within an LMA or MAG.

This document discusses various deployment options for mobility management in a Control-/Data-Plane separated setup utilizing SDN and how results of the chartered work about Forwarding Policy Control (FPC) [I-D.ietf-dmm-fpc-cpdp] can be used in such setup.

The diagrams and descriptions as per this document are meant for discussion and consideration in the DMM working group’s chartered item about DMM deployment options.

The WG document [I-D.ietf-dmm-fpc-cpdp] specifies an abstraction and information model for DMM control and data plane separation. The specified functional architecture comprises a Client function, which is associated with the mobility management Control-Plane, and an Agent function, which is associated with Data-Plane configuration. Client and Agent interact with each other as per the specification in [I-D.ietf-dmm-fpc-cpdp] to enforce forwarding and policy rules from the mobility management Control-Plane in the Data-Plane. Such rules comprise, for example, forwarding of defined traffic through a GRE tunnel to a remote Data-Plane node while the traffic should experience prioritization as per a defined Quality-of-Service class. Such interaction is illustrated in Figure 1.
In the present document we describe a base reference model from [I-D.ietf-dmm-fpc-cpdp], to then propose some sub-scenarios of interest for practical deployment. In the base reference model, as depicted in Figure 2, the Control-/Data-Plane interaction is performed through an NC that exhibits a southbound interface (SBI), e.g. the OpenFlow protocol, to configure DPNs, and exposes a northbound interface (NBI) towards mobility management Control-Plane functions utilizing the Client and Agent functions as well as the abstraction and configuration model as per [I-D.ietf-dmm-fpc-cpdp]. This enables the mobility management Control-Plane using the NC’s NBI to enforce rules and policies in the Data-Plane, which are relevant for the Control-Plane’s operation, without the need to be aware of a Data-Plane node’s configuration details and whether the Data-Plane node is a switch or a router.

An SDN-based Control- and Data-plane separation for DMM has been proposed also in [I-D.yang-dmm-sdn-dmm]. In such draft the mobility function is co-located with the SDN Network Controller. By doing so, the mobility protocol is highly coupled with the NC and the SBI interface. The present draft extends [I-D.yang-dmm-sdn-dmm] by introducing the use of an NBI as an additional abstraction layer. In this way the architecture gains in flexibility as the mobility function is independent from the specific SBI and NC used. The NBI hides Data-Plane configuration details from Control-Plane functions and is represented by a protocol implementation as per [I-D.ietf-dmm-fpc-cpdp].
The flexibility introduced by splitting the Control-/Data-Plane reference point into two different interfaces, namely the NBI and the SBI, may lead to practical limitations for a single network controller to handle the whole set of data plane nodes. As a consequence, when multiple NCs are to be deployed, different scenarios arise with different characteristics. This document explores three possible deployment scenarios.

2. Terminology

The following terms are defined and used in this document:

FPCP (Forwarding Policy Configuration Protocol)

MCF (Mobility Control Function)

NC (Network Controller)

NBI (North-bound Interface)

SBI (South-bound Interface)

DPN (Data Plane Node)

MN (Mobile Node)
3. Control plane and data plane separation: deployment scenarios

A single network controller may lead to practical limitations when it comes to handle the operations in large network, like scalability issues and protocol promptness.

The mobility management protocol should execute quickly in order to guarantee optimal service to users. For this reason the network controller should not be too distant from the DPNs. Therefore, a network administrator might deploy multiple NCs, each responsible for a network sub-domain or site.

By deploying multiple NCs, the Control-/Data-Plane reference model as depicted in Figure 2 suffers some variations as described in the following subsections.

3.1. Scenario 1

In this scenario the NCs from different sites are directly interacting with the mobility control function by means of the NBI interface. The mobility control function possesses a holistic view of the network domain and propagates the instruction to the NCs accordingly. The mobility control function, which is associated with the Client function as per [I-D.ietf-dmm-fpc-cpdp], connects to two Agent functions, each associated with one NC. This scenario is illustrated in Figure 3.

![Diagram of Scenario 1](image-url)

Figure 3: Scenario 1
3.2. Scenario 2

In this scenario, the mobility control Client function interacts with a single Agent co-located with one NC, which in turns propagates the configuration through an east-west interface (EWI) to other NCs. This scenario is illustrated in Figure 4.

![Diagram of Scenario 2]

Figure 4: Scenario 2

3.3. Scenario 3

In this scenario, each NC interacts through an Agent with a dedicated instance of the mobility function Client. It is then up to the different mobility control entities to coordinate the operations among different sites. This scenario is illustrated in Figure 5.

![Diagram of Scenario 3]
4. References

4.1. Normative References


4.2. Informative References


[I-D.yang-dmm-sdn-dmm] yangun@dcn.ssu.ac.kr, y. and Y. Kim, "Routing Optimization with SDN", draft-yang-dmm-sdn-dmm-03 (work in progress), March 2015.

Authors' Addresses
Protocol for Forwarding Policy Configuration (FPC) in DMM
draft-ietf-dmm-fpc-cpdp-12

Abstract

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

Status of This Memo

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 https://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 December 20, 2018.
1. Introduction

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

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

A FPC agent represents the data-plane operation according to several basic information models. A FPC agent also provides access to Monitors, which produce reports when triggered by events or FPC Client requests regarding Mobility Contexts, DPNs or the Agent.
To manage mobility sessions, the FPC client assembles applicable sets of forwarding policies from the data model, and configures them on the appropriate FPC Agent. The Agent then renders those policies into specific configurations for each DPN at which mobile nodes are attached. The specific protocols and configurations to configure a DPN from a FPC Agent are outside the scope of this document.

A DPN is a logical entity that performs data-plane operations (packet movement and management). It may represent a physical DPN unit, a sub-function of a physical DPN or a collection of physical DPNs (i.e., a "virtual DPN"). A DPN may be virtual -- it may export the FPC DPN Agent interface, but be implemented as software that controls other data-plane hardware or modules that may or may not be FPC-compliant. In this document, DPNs are specified without regard for whether the implementation is virtual or physical. DPNs are connected to provide mobility management systems such as access networks, anchors and domains. The FPC agent interface enables establishment of a topology for the forwarding plane.

When a DPN is mapped to physical data-plane equipment, the FPC client can have complete knowledge of the DPN architecture, and use that information to perform DPN selection for specific sessions. On the other hand, when a virtual DPN is mapped to a collection of physical DPNs, the FPC client cannot select a specific physical DPN because it is hidden by the abstraction; only the FPC Agent can address the specific associated physical DPNs. Network architects have the flexibility to determine which DPN-selection capabilities are performed by the FPC Agent (distributed) and which by the FPC client (centralized). In this way, overlay networks can be configured without disclosing detailed knowledge of the underlying hardware to the FPC client and applications.

The abstractions in this document are designed to support many different mobility management systems and data-plane functions. The architecture and protocol design of FPC is not tied to specific types of access technologies and mobility protocols.

2. Terminology

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

Attribute Expression: The definition of a template Property. This includes setting the type, current value, default value and if the attribute is static, i.e. can no longer be changed.
Domain: One or more DPNs that form a logical partition of network resources (e.g., a data-plane network under common network administration). A FPC client (e.g., a mobility management system) may utilize a single or multiple domains.

DPN: A data-plane node (DPN) is capable of performing data-plane features. For example, DPNs may be switches or routers, regardless of whether they are realized as hardware or purely in software.

FPC Client: A FPC Client is integrated with a mobility management system or related application, enabling control over forwarding policy, mobility sessions and DPNs via a FPC Agent.

Mobility Context: A Mobility Context contains the data-plane information necessary to efficiently send and receive traffic from a mobile node. This includes policies that are created or modified during the network’s operation – in most cases, on a per-flow or per session basis. A Mobility-Context represents the mobility sessions (or flows) which are active on a mobile node. This includes associated runtime attributes, such as tunnel endpoints, tunnel identifiers, delegated prefix(es), routing information, etc. Mobility-Contexts are associated to specific DPNs. Some pre-defined Policies may apply during mobility signaling requests. The Mobility Context supplies information about the policy settings specific to a mobile node and its flows; this information is often quite dynamic.

Mobility Session: Traffic to/from a mobile node that is expected to survive reconnection events.

Monitor: A reporting mechanism for a list of events that trigger notification messages from a FPC Agent to a FPC Client.

Policy: A Policy determines the mechanisms for managing specific traffic flows or packets. Policies specify QoS, rewriting rules for
packet processing, etc. A Policy consists of one or more rules. Each rule is composed of a Descriptor and Actions. The Descriptor in a rule identifies packets (e.g., traffic flows), and the Actions apply treatments to packets that match the Descriptor in the rule. Policies can apply to Domains, DPNs, Mobile Nodes, Service-Groups, or particular Flows on a Mobile Node.

Property: An attribute-value pair for an instance of a FPC entity.

Service-Group: A set of DPN interfaces that support a specific data-plane purpose, e.g. inbound/outbound, roaming, subnetwork with common specific configuration, etc.

Template: A recipe for instantiating FPC entities. Template definitions are accessible (by name or by a key) in an indexed set. A Template is used to create specific instances (e.g., specific policies) by assigning appropriate values into the Template definition via Attribute Expression.

Template Configuration The process by which a Template is referenced (by name or by key) and Attribute Expressions are created that change the value, default value or static nature of the Attribute, if permitted. If the Template is Extensible, new attributes MAY be added.

Tenant: An operational entity that manages mobility management systems or applications which require data-plane functions. A Tenant defines a global namespace for all entities owned by the Tenant enabling its entities to be used by multiple FPC Clients across multiple FPC Agents.

Topology: The DPNs and the links between them. For example, access nodes may be assigned to a Service-Group which peers to a Service-Group of anchor nodes.
3. FPC Design Objectives and Deployment

Using FPC, mobility control-planes and applications can configure DPNs to perform various mobility management roles as described in [I-D.ietf-dmm-deployment-models]. This fulfills the requirements described in [RFC7333].

This document defines FPC Agent and FPC Client, as well as the information models that they use. The attributes defining those models serve as the protocol elements for the interface between the FPC Agent and the FPC Client.

Mobility control-plane applications integrate features offered by the FPC Client. The FPC Client connects to FPC Agent functions. The Client and the Agent communicate based on information models described in Section 4. The models allow the control-plane to configure forwarding policies on the Agent for data-plane communications with mobile nodes.

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

A FPC Agent configures and manages its DPN(s) according to forwarding policies requested and Attributes provided by the FPC Client. Configuration commands used by the FPC agent to configure its DPN node(s) may be specific to the DPN implementation; consequently the method by which the FPC Agent carries out the specific configuration for its DPN(s) is out of scope for this document. Along with the data models, the FPC Client (on behalf of control-plane and applications) requests that the Agent configures Policies prior to the time when the DPNs start forwarding data for their mobility sessions.

This architecture is illustrated in Figure 1. A FPC Agent may be implemented in a network controller that handles multiple DPNs, or (more simply) an FPC Agent may itself be integrated into a DPN.

This document does not specify a protocol for the FPC interface; it is out of scope. However, an implementation must support the FPC transactions described in Section 5.
The FPC architecture supports multi-tenancy; a FPC enabled data-plane supports tenants of multiple mobile operator networks and/or applications. It means that the FPC Client of each tenant connects to the FPC Agent and it MUST partition namespace and data for their data-planes. DPNs on the data-plane may fulfill multiple data-plane roles which are defined per session, domain and tenant.
Multi-tenancy permits the partitioning of data-plane entities as well as a common namespace requirement upon FPC Agents and Clients when they use the same Tenant for a common data-plane entity.

FPC information models often configuration to fit the specific needs for DPN management of a mobile node’s traffic. The FPC interfaces in Figure 1 are the only interfaces required to handle runtime data in a Mobility Context. The Topology and some Policy FPC models MAY be pre-configured; in that case real-time protocol exchanges are not required for them.

The information model provides an extensibility mechanism through Templates that permits specialization for the needs of a particular vendor’s equipment or future extension of the model presented in this specification.

4. FPC Mobility Information Model

The FPC information model includes the following components:

DPN Information Model,
Topology Information Model,
Policy Information Model,
Mobility-Context, and
Monitor, as illustrated in Figure 2.

```
  +-[FPC Mobility Information Model]
    |                                    
    +-[Topology Information Model]      
    |                                     
    +-[Policy Information Model]        
    |                                     
    +-[Mobility-Context]                
    |                                     
    +-[Monitor]                          
```

Figure 2: FPC Information Model structure

4.1. Model Notation and Conventions

The following conventions are used to describe the FPC information models.

Information model entities (e.g. DPNs, Rules, etc.) are defined in a hierarchical notation where all entities at the same hierarchical
level are located on the same left-justified vertical position sequentially. When entities are composed of sub-entities, the sub-entities appear shifted to the right, as shown in Figure 3.

```
  +-[entity2]
    +-[entity2.1]
    +-[entity2.2]
```

Figure 3: Model Notation - An Example

Some entities have one or more qualifiers placed on the right hand side of the element definition in angle-brackets. Common types include:

- **List**: A collection of entities (some could be duplicated)
- **Set**: A nonempty collection of entities without duplications
- **Name**: A human-readable string
- **Key**: A unique value. We distinguish 3 types of keys:
  - **U-Key**: A key unique across all Tenants. U-Key spaces typically involve the use of registries or language specific mechanisms that guarantee universal uniqueness of values.
  - **G-Key**: A key unique within a Tenant
  - **L-Key**: A key unique within a local namespace. For example, there may exist interfaces with the same name, e.g. "if0", in two different DPNs but there can only be one "if0" within each DPN (i.e. its local Interface-Key L-Key space).

Each entity or attribute may be optional (O) or mandatory (M). Entities that are not marked as optional are mandatory.
The following example shows 3 entities:

-- Entity1 is a globally unique key, and optionally can have an associated Name
-- Entity2 is a list
-- Entity3 is a set and is optional

\[
\begin{align*}
\text{+} \\
\text{|} \\
\text{+-[entity1] <G-Key> (M), <Name> (O)} \\
\text{+-[entity2] <List>} \\
\text{+-[entity3] <Set> (O)} \\
\text{+}
\end{align*}
\]

Figure 4

When expanding entity1 into a modeling language such as YANG it would result in two values: entity1-Key and entity1-Name.

To encourage re-use, FPC defines indexed sets of various entity Templates. Other model elements that need access to an indexed model entity contain an attribute which is always denoted as "entity-Key". When a Key attribute is encountered, the referencing model element may supply attribute values for use when the referenced entity model is instantiated. For example: Figure 5 shows 2 entities:

EntityA definition references an entityB model element.

EntityB model elements are indexed by entityB-Key.

Each EntityB model element has an entityB-Key which allows it to be uniquely identified, and a list of Attributes (or, alternatively, a Type) which specifies its form. This allows a referencing entity to create an instance by supplying entityB-Values to be inserted, in a Settings container.
Figure 5: Indexed sets of entities

Indexed sets are specified for each of the following kinds of entities:

- Domain (See Section 4.9.3)
- DPN (See Section 4.9.4)
- Policy (See Section 4.9.5)
- Rule (See Section 4.9.5)
- Descriptor (See Figure 12)
- Action (See Figure 12)
- Service-Group (See Section 4.9.2, and
- Mobility-Context (See Section 4.9.6)

As an example, for a Domain entity, there is a corresponding attribute denoted as "Domain-Key" whose value can be used to determine a reference to the Domain.

4.2. Templates and Attributes

In order to simplify development and maintenance of the needed policies and other objects used by FPC, the Information Models which are presented often have attributes that are not initialized with their final values. When an FPC entity is instantiated according to a template definition, specific values need to be configured for each such attribute. For instance, suppose an entity Template has an Attribute named "IPv4-Address", and also suppose that a FPC Client instantiates the entity and requests that it be installed on a DPN. An IPv4 address will be needed for the value of that Attribute before the entity can be used.
Figure 6: Template entities

Attributes: A set of Attribute names MAY be included when defining a Template for instantiating FPC entities.

Extensible: Determines whether or not entities instantiated from the Template can be extended with new non-mandatory Attributes not originally defined for the Template. Default value is FALSE. If a Template does not explicitly specify this attribute, the default value is considered to be in effect.

Entity-State: Either Initial, PartiallyConfigured, Configured, or Active. Default value is Initial. See Section 4.6 for more information about how the Entity-Status changes during the configuration steps of the Entity.

Version: Provides a version tag for the Template.

The Attributes in an Entity Template may be either mandatory or non-mandatory. Attribute values may also be associated with the attributes in the Entity Template. If supplied, the value may be either assigned with a default value that can be reconfigured later, or the value can be assigned with a static value that cannot be reconfigured later (see Section 4.3).

It is possible for a Template to provide values for all of its Attributes, so that no additional values are needed before the entity can made Active. Any instantiation from a Template MUST have at least one Attribute in order to be a useful entity unless the Template has none.

4.3. Attribute-Expressions

The syntax of the Attribute definition is formatted to make it clear. For every Attribute in the Entity Template, six possibilities are specified as follows:

’[Att-Name: ]’ Mandatory Attribute is defined, but template does not provide any configured value.

’[Att-Name: Att-Value]’ Mandatory Attribute is defined, and has a statically configured value.
'Mandatory Attribute is defined, and has a
default value.

'Mandatory Attribute may be included but template
does not provide any configured value.

Non-mandatory Attribute may be included and
has a statically configured value.

Non-mandatory Attribute may be included and
has a default value.

So, for example, a default value for a non-mandatory IPv4-Address
attribute would be denoted by [IPv4-Address ˜ 127.0.0.1].

After a FPC Client identifies which additional Attributes have been
configured to be included in an instantiated entity, those configured
Attributes MUST NOT be deleted by the FPC Agent. Similarly, any
statically configured value for an entity Attribute MUST NOT be
changed by the FPC Agent.

Whenever there is danger of confusion, the fully qualified Attribute
name MUST be used when supplying needed Attribute Values for a
structured Attribute.

4.4. Attribute Value Types

For situations in which the type of an attribute value is required,
the following syntax is recommended. To declare than an attribute
has data type "foo", typecast the attribute name by using the
parenthesized data type (foo). So, for instance, [(float) Max-
Latency-in-ms:] would indicate that the mandatory Attribute "Max-
Latency-in-ms" requires to be configured with a floating point value
before the instantiated entity could be used. Similarly, [(float)
Max-Latency-in-ms: 9.5] would statically configure a floating point
value of 9.5 to the mandatory Attribute "Max-Latency-in-ms".

4.5. Namespace and Format

The identifiers and names in FPC models which reside in the same
Tenant must be unique. That uniqueness must be maintained by all
Clients, Agents and DPNs that support the Tenant. The Tenant
namespace uniqueness MUST be applied to all elements of the tenant
model, i.e. Topology, Policy and Mobility models.

When a Policy needs to be applied to Mobility-Contexts in all Tenants
on an Agent, the Agent SHOULD define that policy to be visible by all
Tenants. In this case, the Agent assigns a unique identifier in the
Agent namespace and copies the values to each Tenant. This effectively creates a U-Key although only a G-Key is required within the Tenant.

The notation for identifiers can utilize any format with agreement between data-plane agent and client operators. The formats include but are not limited to Globally Unique IDentifiers (GUIDs), Universally Unique IDentifiers (UUIDs), Fully Qualified Domain Names (FQDNs), Fully Qualified Path Names (FQPNs) and Uniform Resource Identifiers (URIs). The FPC model does not limit the format, which could dictate the choice of FPC protocol. Nevertheless, the identifiers which are used in a Mobility model should be considered to efficiently handle runtime parameters.

There are identifiers reserved for Protocol Operation. See Section 5.1.1.5 for details.

4.6. Configuring Attribute Values

Attributes of Information Model components such as policy templates are configured with values as part of FPC configuration operations. There may be several such configuration operations before the template instantiation is fully configured.

Entity-Status indicates when an Entity is usable within a DPN. This permits DPN design tradeoffs amongst local storage (or other resources), over the wire request size and the speed of request processing. For example, DPN designers with constrained systems MAY only house entities whose status is Active which may result in sending over all policy information with a Mobility-Context request. Storing information elements with an entity status of "PartiallyConfigured" on the DPN requires more resources but can result in smaller over the wire FPC communication and request processing efficiency.

When the FPC Client instantiates a Policy from a Template, the Policy-Status is "Initial". When the FPC Client sends the policy to a FPC Agent for installation on a DPN, the Client often will configure appropriate attribute values for the installation, and accordingly changes the Policy-Status to "PartiallyConfigured" or "Configured". The FPC Agent will also configure Domain-specific policies and DPN-specific policies on the DPN. When configured to provide particular services for mobile nodes, the FPC Agent will apply whatever service-specific policies are needed on the DPN. When a mobile node attaches to the network data-plane within the topology under the jurisdiction of a FPC Agent, the Agent may apply policies and settings as appropriate for that mobile node. Finally, when the mobile node launches new flows, or quenches existing flows, the FPC
Agent, on behalf of the FPC Client, applies or deactivates whatever policies and attribute values are appropriate for managing the flows of the mobile node. When a "Configured" policy is de-activated, Policy-Status is changed to be "Active". When an "Active" policy is activated, Policy-Status is changed to be "Configured".

Attribute values in DPN resident Policies may be configured by the FPC Agent as follows:

Domain-Policy-Configuration: Values for Policy attributes that are required for every DPN in the domain.

DPN-Policy-Configuration: Values for Policy attributes that are required for every policy configured on this DPN.

Service-Group-Policy-Configuration: Values for Policy attributes that are required to carry out the intended Service of the Service Group.

MN-Policy-Configuration: Values for Policy attributes that are required for all traffic to/from a particular mobile node.

Service-Data-Flow-Policy-Configuration: Values for Policy attributes that are required for traffic belonging to a particular set of flows on the mobile node.

Any configuration changes MAY also supply updated values for existing default attribute values that may have been previously configured on the DPN resident policy.

Entity blocks describe the format of the policy configurations.

4.7. Entity Configuration Blocks

As described in Section 4.6, a Policy Template may be configured in several stages by configuring default or missing values for Attributes that do not already have statically configured values. A Policy-Configuration is the combination of a Policy-Key (to identify the Policy Template defining the Attributes) and the currently configured Attribute Values to be applied to the Policy Template. Policy-Configurations MAY add attributes to a Template if Extensible is True. They MAY also refine existing attributes by:

- assign new values if the Attribute is not static
- make attributes static if they were not
- make an attribute mandatory
A Policy-Configuration MUST NOT define or refine an attribute twice. More generally, an Entity-Configuration can be defined for any configurable Indexed Set to be the combination of the Entity-Key along with a set of Attribute-Expressions that supply configuration information for the entity’s Attributes. Figure 7 shows a schematic representation for such Entity Configuration Blocks.

![Entity Configuration Block Diagram]

Figure 7: Entity Configuration Block

This document makes use of the following kinds of Entity Configuration Blocks:

- Descriptor-Configuration
- Action-Configuration
- Rule-Configuration
- Interface-Configuration
- Service-Group-Configuration
- Domain-Policy-Configuration
- DPN-Policy-Configuration
- Policy-Configuration
- MN-Policy-Configuration
- Service-Data-Flow-Policy-Configuration

4.8. Information Model Checkpoint

The Information Model Checkpoint permits Clients and Tenants with common scopes, referred to in this specification as Checkpoint BaseNames, to track the state of provisioned information on an Agent. The Agent records the Checkpoint BaseName and Checkpoint value set by a Client. When a Client attaches to the Agent it can query to determine the amount of work that must be executed to configure the Agent to a specific BaseName / checkpoint revision.

Checkpoints are defined for the following information model components:
Service-Group

DPN Information Model

Domain Information Model

Policy Information Model

4.9. Information Model Components

4.9.1. Topology Information Model

The Topology structure specifies DPNs and the communication paths between them. A network management system can use the Topology to select the most appropriate DPN resources for handling specific session flows.

The Topology structure is illustrated in Figure 8 (for definitions see Section 2):

```
|--[Topology Information Model]
   |--[Extensible: FALSE]
   |   |--[Service-Group]
   |   |   |--[DPN] <Set>
   |   |   |--[Domain] <Set>
```

Figure 8: Topology Structure

4.9.2. Service-Group

Service-Group-Set is collection of DPN interfaces serving some data-plane purpose including but not limited to DPN Interface selection to fulfill a Mobility-Context. Each Group contains a list of DPNs (referenced by DPN-Key) and selected interfaces (referenced by Interface-Key). The Interfaces are listed explicitly (rather than referred implicitly by its specific DPN) so that every Interface of a DPN is not required to be part of a Group. The information provided is sufficient to ensure that the Protocol, Settings (stored in the Service-Group-Configuration) and Features relevant to successful interface selection is present in the model.
Figure 9: Service Group

Each Service-Group element contains the following information:

Service-Group-Key: A unique ID of the Service-Group.

Service-Group-Name: A human-readable display string.

Role: The role (MAG, LMA, etc.) of the device hosting the interfaces of the DPN Group.

Protocol-Set: The set of protocols supported by this interface (e.g., PMIP, S5-GTP, S5-PMIP etc.). The protocol MAY be only its name, e.g. ‘gtp’, but many protocols implement specific message sets, e.g. s5-pmip, s8-pmip. When the Service-Group supports specific protocol message sub-subsets the Protocol value MUST include this information.

Feature-Set: An optional set of static features which further determine the suitability of the interface to the desired operation.

Service-Group-Configuration-Set: An optional set of configurations that further determine the suitability of an interface for the specific request. For example: SequenceNumber=ON/OFF.

DPN-Key-Set: A key used to identify the DPN.

Referenced-Interface-Set: The DPN Interfaces and peer Service-Groups associated with them. Each entry contains

Interface-Key: A key that is used together with the DPN-Key, to create a key that is refers to a specific DPN interface definition.
Peer-Service-Group-Key: Enables location of the peer Service-Group for this Interface.

4.9.3. Domain Information Model

A Domain-Set represents a group of heterogeneous Topology resources typically sharing a common administrative authority. Other models, outside of the scope of this specification, provide the details for the Domain.

```
+[Domain] <G-Key>, <Name> (O) <Set>
 |    +-[Domain-Policy-Configuration] (O) <Set>
```

Figure 10: Domain Information Model

Each Domain entry contains the following information:

Domain-Key: Identifies and enables reference to the Domain.

Domain-Name: A human-readable display string naming the Domain.

4.9.4. DPN Information Model

A DPN-Set contains some or all of the DPNs in the Tenant’s network. Some of the DPNs in the Set may be identical in functionality and only differ by their Key.

```
+[DPN] <G-Key>, <Name> (O) <Set>
 +-[Extensible: FALSE]
   +-[Interface] <L-Key> <Set>
     +-[Role] <U-Key>
     +-[Protocol] <Set>
     +-[Interface-Configuration] <Set> (O)
    +-[Domain-Key]
    +-[Service-Group-Key] <Set> (O)
    +-[DPN-Policy-Configuration] <List> (M)
    +-[DPN-Resource-Mapping-Reference] (O)
```

Figure 11: DPN Information Model

Each DPN entry contains the following information:

DPN-Key: A unique Identifier of the DPN.

DPN-Name: A human-readable display string.
Domain-Key: A Key providing access to the Domain information about the Domain in which the DPN resides.

Interface-Set: The Interface-Set references all interfaces (through which data packets are received and transmitted) available on the DPN. Each Interface makes use of attribute values that are specific to that interface, for example, the MTU size. These do not affect the DPN selection of active or enabled interfaces. Interfaces contain the following information:

Role: The role (MAG, LMA, PGW, AMF, etc.) of the DPN.

Protocol (Set): The set of protocols supported by this interface (e.g., PMIP, S5-GTP, S5-PMIP etc.). The protocol MAY implement specific message sets, e.g. s5-pmip, s8-pmip. When a protocol implements such message sub-subsets the Protocol value MUST include this information.

Interface-Configuration-Set: Configurable settings that further determine the suitability of an interface for the specific request. For example: SequenceNumber=ON/OFF.

Service-Group-Set: The Service-Group-Set references all of the Service-Groups which have been configured using Interfaces hosted on this DPN. The purpose of a Service-Group is not to describe each interface of each DPN, but rather to indicate interface types for use during the DPN selection process, when a DPN with specific interface capabilities is required.

DPN-Policy-Configuration: A list of Policies that have been configured on this DPN. Some may have values for all attributes, and some may require further configuration. Each Policy-Configuration has a key to enable reference to its Policy-Template. Each Policy-Configuration also has been configured to supply missing and non-default values to the desired Attributes defined within the Policy-Template.

DPN-Resource-Mapping-Reference (O): A reference to the underlying implementation, e.g. physical node, software module, etc. that supports this DPN. Further specification of this attribute is out of scope for this document.

4.9.5. Policy Information Model

The Policy Information Model defines and identifies Rules for enforcement at DPNs. A Policy is basically a set of Rules that are to be applied to each incoming or outgoing packet at a DPN interface. Rules comprise Descriptors and a set of Actions. The Descriptors,
when evaluated, determine whether or not a set of Actions will be performed on the packet. The Policy structure is independent of a policy context.

In addition to the Policy structure, the Information Model (per Section 4.9.6) defines Mobility-Context. Each Mobility-Context may be configured with appropriate Attribute values, for example depending on the identity of a mobile node.

Traffic descriptions are defined in Descriptors, and treatments are defined separately in Actions. A Rule-Set binds Descriptors and associated Actions by reference, using Descriptor-Key and Action-Key. A Rule-Set is bound to a policy in the Policy-Set (using Policy-Key), and the Policy references the Rule definitions (using Rule-Key).

```
+-[Policy Information Model]
  | +-[Extensible:]
  |   +-[Policy-Template] <G-Key> (M) <Set>
  |       +-[Policy-Configuration] <Set> (O)
  |       +-[Rule-Template-Key] <List> (M)
  |          +-[Precedence] (M)
  |   +-[Rule-Template] <L-Key> (M) <Set>
  |       +-[Descriptor-Match-Type] (M)
  |       +-[Descriptor-Configuration] <Set> (M)
  |          +-[Direction] (O)
  |       +-[Action-Configuration] <Set> (M)
  |          +-[Action-Order] (M)
  |   +-[Rule-Configuration] (O)
  |   +-[Descriptor-Template] <L-Key> (M) <Set>
  |       +-[Descriptor-Type] (O)
  |       +-[Attribute-Expression] <Set> (M)
  |   +-[Action-Template] <L-Key> (M) <Set>
  |       +-[Action-Type] (O)
  |          +-[Attribute-Expression] <Set> (M)
```

Figure 12: Policy Information Model

The Policy structure defines Policy-Set, Rule-Set, Descriptor-Set, and Action-Set, as follows:

Policy-Template: <Set> A set of Policy structures, indexed by Policy-Key, each of which is determined by a list of Rules referenced by their Rule-Key. Each Policy structure contains the following:

Policy-Key: Identifies and enables reference to this Policy definition.
Rule-Template-Key: Enables reference to a Rule template definition.

Rule-Precedence: For each Rule identified by a Rule-Template-Key in the Policy, specifies the order in which that Rule must be applied. The lower the numerical value of Precedence, the higher the rule precedence. Rules with equal precedence MAY be executed in parallel if supported by the DPN. If this value is absent, the rules SHOULD be applied in the order in which they appear in the Policy.

Rule-Template-Set: A set of Rule Template definitions indexed by Rule-Key. Each Rule is defined by a list of Descriptors (located by Descriptor-Key) and a list of Actions (located by Action-Key) as follows:

Rule-Template-Key: Identifies and enables reference to this Rule definition.

Descriptor-Match-Type: Indicates whether the evaluation of the Rule proceeds by using conditional-AND, or conditional-OR, on the list of Descriptors.

Descriptor-Configuration: References a Descriptor template definition, along with an expression which names the Attributes for this instantiation from the Descriptor-Template and also specifies whether each Attribute of the Descriptor has a default value or a statically configured value, according to the syntax specified in Section 4.2.

Direction: Indicates if a rule applies to uplink traffic, to downlink traffic, or to both uplink and downlink traffic. Applying a rule to both uplink and downlink traffic, in case of symmetric rules, eliminates the requirement for a separate entry for each direction. When not present, the direction is implied by the Descriptor’s values.

Action-Configuration: References an Action Template definition, along with an expression which names the Attributes for this instantiation from the Action-Template and also specifies whether each Attribute of the Action has a default value or a statically configured value, according to the syntax specified in Section 4.2.

Action-Order: Defines the order in which actions are executed when the associated traffic descriptor selects the packet.
Descriptor-Template-Set: A set of traffic Descriptor Templates, each of which can be evaluated on the incoming or outgoing packet, returning a TRUE or FALSE value, defined as follows:

Descriptor-Template-Key: Identifies and enables reference to this descriptor template definition.

Attribute-Expression: An expression which defines an Attribute in the Descriptor-Template and also specifies whether the Template also defines a default value or a statically configured value for the Attribute of the Descriptor has, according to the syntax specified in Section 4.2.

Descriptor-Type: Identifies the type of descriptor, e.g. an IPv6 traffic selector per [RFC6088].

Action-Template-Set: A set of Action Templates defined as follows:

Action-Template-Key: Identifies and enables reference to this action template definition.

Attribute-Expression: An expression which defines an Attribute in the Action-Template and also specifies whether the Template also defines a default value or a statically configured value for the Attribute of the Action has, according to the syntax specified in Section 4.2.

Action-Type: Identifies the type of an action for unambiguous interpretation of an Action-Value entry.

4.9.6. Mobility-Context Information Model

The Mobility-Context structure holds entries associated with a mobile node and its mobility sessions (flows). It is created on a DPN during the mobile node’s registration to manage the mobile node’s flows. Flow information is added or deleted from the Mobility-Context as needed to support new flows or to deallocate resources for flows that are deactivated. Descriptors are used to characterize the nature and resource requirement for each flow.

Termination of a Mobility-Context implies termination of all flows represented in the Mobility-Context, e.g. after deregistration of a mobile node. If any Child-Contexts are defined, they are also terminated.
The Mobility-Context Substructure holds the following entries:

**Mobility-Context-Key**: Identifies a Mobility-Context

**Delegating-IP-Prefix-Set**: Delegated IP Prefixes assigned to the Mobility-Context

**Parent-Context**: If present, a Mobility Context from which the Attributes and Attribute Values of this Mobility Context are inherited.

**Child-Context-Set**: A set of Mobility Contexts which inherit the Attributes and Attribute Values of this Mobility Context.

**Service-Group-Key**: Service-Group(s) used during DPN assignment and re-assignment.

**Mobile-Node**: Attributes specific to the Mobile Node. It contains the following

- **IP-Address-Set**: IP addresses assigned to the Mobile Node.

- **MN-Policy-Configuration-Set**: For each MN-Policy in the set, a key and relevant information for the Policy Attributes.
Domain-Key: Enables access to a Domain instance.

Domain-Policy-Configuration-Set: For each Domain-Policy in the set, a key and relevant information for the Policy Attributes.

DPN-Key-Set: Enables access to a DPN instance assigned to a specific role, i.e. this is a Set that uses DPN-Key and Role as a compound key to access specific set instances.

Role: Role this DPN fulfills in the Mobility-Context.

DPN-Policy-Configuration-Set: For each DPN-Policy in the set, a key and relevant information for the Policy Attributes.

ServiceDataFlow-Key-Set: Characterizes a traffic flow that has been configured (and provided resources) on the DPN to support data-plane traffic to and from the mobile device.

Service-Group-Key: Enables access to a Service-Group instance.

Interface-Key-Set: Assigns the selected interface of the DPN.

ServiceDataFlow-Policy-Configuration-Set: For each Policy in the set, a key and relevant information for the Policy Attributes.

Direction: Indicates if the reference Policy applies to uplink or downlink traffic, or to both, uplink- and downlink traffic. Applying a rule to both, uplink- and downlink traffic, in case of symmetric rules, allows omitting a separate entry for each direction. When not present the value is assumed to apply to both directions.

4.9.7. Monitor Information Model

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

The attribute/entity to be monitored places certain constraints on the configuration that can be specified. For example, a Monitor using a Threshold configuration cannot be applied to a Mobility-Context, because it does not have a threshold. Such a monitor configuration could be applied to a numeric threshold property of a Context.
Monitor-Key: Identifies the Monitor.

Target: Description of what is to be monitored. This can be a Service Data Flow, a Policy installed upon a DPN, values of a Mobility-Context, etc. The target name is the absolute information model path (separated by '/') to the attribute / entity to be monitored.

Deferrable: Indicates that a monitoring report can be delayed up to a defined maximum delay, set in the Agent, for possible bundling with other reports.

Configuration: Determined by the Monitor subtype. The monitor report is specified by the Configuration. Four report types are defined:

* "Periodic" reporting specifies an interval by which a notification is sent.

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

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

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

5. Protocol

5.1. Protocol Messages and Semantics

Four Client to Agent messages are supported.
<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure</td>
<td>A Configure message includes multiple edits to one or more information model entities. Edits are executed according to their Edit-Id in ascending order. The global status of the operation and the status of individual edits are returned. Partial failures, i.e. individual edit failures, are allowed.</td>
</tr>
<tr>
<td>Register-Monitors</td>
<td>Register monitors at an Agent. The message includes the Monitor information as specified in Section 4.9.7.</td>
</tr>
<tr>
<td>Deregister-Monitors</td>
<td>Deregister monitors from an Agent. An optional boolean, Send-Data, indicates if a successful deregistration triggers a Notify with final data from the Agent for the corresponding Monitor.</td>
</tr>
<tr>
<td>Probe</td>
<td>Probe the status of registered monitors. This triggers a Notify with current data from the Agent for the corresponding Monitors.</td>
</tr>
</tbody>
</table>

Table 1: Client to Agent Messages

Each message contains a header with the following information:

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

Delay: An optional time (in ms) to delay the execution of the operation on the DPN once it is received by the Agent.

Operation Identifier: A unique identifier created by the Client to correlate responses and notifications.

An Agent will respond with an ERROR, indicating one or more Errors have occurred, or an OK.

For Configure messages, an OK status for an edit MAY include subsequent edits in the response that were required to properly execute the edit. It MAY also indicate that the final status and any final edits required to fulfill the request will be sent via a...
Configure Result Notification from the Agent to the Client, see Section 5.1.1.4.2.

If errors occur, they MUST be returned as a list in responses and each Error contains the following information:

Error-type: The specific error type. Values are TRANSPORT (0), RPC (1), PROTOCOL(2) or APPLICATION (3).

Error-Tag: An error tag.

Error-App-Tag: Application specific error tag.

Error-Message: A message describing the error.

Error-Info: Any data required for the response.

```
+-[Errors] <List>
 | +-[(Enumeration) Error-Type ]
 | +-[(String) Error-Tag ]
 | +-[(String) Error-App-Tag ] (O)
 | +-[(String) Error-Message ] (O)
 | +-[Error-Info] (O)
```

Figure 15: Error Information Model

Two Agent to Client notifications are supported.

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure-Result-Notification</td>
<td>An asynchronous notification from Agent to Client based upon a previous Configure request.</td>
</tr>
<tr>
<td>Notify</td>
<td>An asynchronous notification from Agent to Client based upon a registered Monitor's configuration, a Monitor deregistration or Probe.</td>
</tr>
</tbody>
</table>

Table 2: Agent to Client Messages (notifications)
5.1.1. Configure Message

The Configure message follows edit formats proposed by [RFC8072] with more fields in each edit, an extra operation (clone) and a different response format.

5.1.1.1. Edit Operation Types

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>create</td>
<td>Creates a new data resource or Entity. If the resource exists an error is returned.</td>
</tr>
<tr>
<td>delete</td>
<td>Deletes a resource. If it does not exist an error is returned.</td>
</tr>
<tr>
<td>insert</td>
<td>Inserts data in a list or user ordered list.</td>
</tr>
<tr>
<td>merge</td>
<td>Merges the edit value with the target data resource; the resource is created if it does not exist.</td>
</tr>
<tr>
<td>move</td>
<td>Moves the target data resource.</td>
</tr>
<tr>
<td>replace</td>
<td>Replace the target data resource with the edit value.</td>
</tr>
<tr>
<td>remove</td>
<td>Removes a data resource if it already exists.</td>
</tr>
<tr>
<td>clone</td>
<td>Clones a data resource and places the copy at the new location. If the resource does not exist an error is returned.</td>
</tr>
</tbody>
</table>

Table 3: Configure Edit Operations

5.1.1.2. Edit Operation

Each Configure includes one or more edits. These edits include the following information:

Edit-Id: Uniquely specifies the identifier of the edit within the operation.

Edit-Type: Specifies the type of operation (see Section 5.1.1.1).

Command-Set: The Command-Set is a technology-specific bitset that allows for a single entity to be sent in an edit with multiple requested, technology specific sub-transactions to be completed. It can also provide clarity for a request. For example, a Mobility-Context could have the Home Network Prefix absent but it is unclear if the Client would like the address to be assigned by the Agent or if this is an error. Rather than creating a specific command for assigning the IP, a bit position in a Command-Set can be used to indicate Agent based IP assignment requests.
Reference-Scope: If supported, specifies the Reference Scope (see Section 5.1.1.3)

Target: Specifies the Target node (Data node path or FPC Identity) for the edit operation. This MAY be a resource, e.g. Mobility-Context, Descriptor-Template, etc., or a data node within a resource as specified by its path.

Point: The absolute URL path for the data node that is being used as the insertion point, clone point or move point for the target of this ‘edit’ entry.

Where: Identifies where a data resource will be inserted, cloned to or moved. Only allowed these for lists and lists of data nodes that are ‘ordered-by user’. The values are ‘before’, ‘after’, ‘first’, ‘last’ (default value).

Value The value used for this edit operation. In this message it MUST NOT be a MONITOR entity.

```
+-[Configure]
  +-[Client-Id:]
  +-[(Unsigned 32) Execution-Delay]
  +-[Operation-Id:]
  +-[Edit:] <List>
    +-[Edit-Id:] <L-Key>
    +-[(Enumeration) Edit-Type:]
    +-[(BitSet) Command-Set]
    +-[(Enumeration) Reference-Scope]
    +-[Target:]
    +-[Point]
    +-[(Enumeration) Where]
    +-[Value]
```

Figure 16: Configure Request

Edits sent to the Agent provided in an operation SHOULD be sent in the following order to avoid errors:

1. Action Templates
2. Descriptor Templates
3. Rule Templates
4. Policy Templates
5. DPN Templates

6. Mobility Contexts

5.1.1.3. Reference Scope

The Reference Scope is an optional feature that provides the scope of references used in a configuration command. These scopes are defined as:

- **none** - All edits have no references to other entities or within edits.
- **edit** - All references are contained within each edit body (intra-edit/intra-operation)
- **operation** - All references exist in the operation (inter-edit/ intra-operation).
- **storage** - One or more references exist outside of the operation. A lookup to cache / storage is required.
- **unknown** - The location of the references are unknown. This is treated as a 'storage' type.

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

Figure 17 shows an example containment hierarchy provided for all caches.
5.1.1.4. Operation Response

5.1.1.4.1. Immediate Response

The Response MUST include the following:

- Operation Identifier of the corresponding request.
- Global Status for the operation (see Table 1).
- A list of Edit results (described below).

An edit response, Edit-Status, is comprised of the following:

- Edit-Id: Edit Identifier.
- Edit-Status: OK.

When the Edit-Status is OK the following values MAY be present:

- Notify-Follows - A boolean indicator that the edit has been accepted by the Agent but further processing is required. A Configure-Result-Notification will be sent once the processing has succeeded or failed.
Subsequent-Edits-List: This is a list of Edits that were required to fulfill the request. It follows the edit request semantics (see Section 5.1.1.2).

Errors-List: When the Edit-Status is ERROR the following values are present. See Table 1 for details.

The response will minimally contain an Edit-Status implying 'OK' or a list of errors.

```markdown
+-[Operation-Id:]
+-[Result-Status:]
+-[Errors] <List>
  +-[{Enumeration} Error-Type:]
  +-[{String} Error-Tag:]
  +-[{String} Error-App-Tag]
  +-[{String} Error-Message]
  +-[Error-Info]
+-[Edit-Status]
  +-[Edit-Id:]
  +-[Edit-Status: OK]
  +-[Notify-Follows]
  +-[Subsequent-Edits] <List>
    +-[Edit-Id:] <L-Key>
      +-[{Enumeration} Edit-Type:]
      +-[Target:]
      +-[Point]
      +-[{Enumeration} Where]
      +-[Value]
    +-[Errors] <List>
      +-[{Enumeration} Error-Type:]
      +-[{String} Error-Tag:]
      +-[{String} Error-App-Tag]
      +-[{String} Error-Message]
      +-[Error-Info]
```

Figure 18: Configure Operation Response

5.1.1.4.2. Asynchronous Notification

A Configure-Result-Notification occurs after the Agent has completed processing related to a Configure request. It is an asynchronous communication from the Agent to the Client.
It is identical to the immediate response with the exception that the Notify-Follows, if present, MUST be false. As this value is unnecessary it SHOULD be omitted.

5.1.1.5. Reserved Identities

Several identities are reserved in the Policy Information Model and Mobility-Context to facilitate specific use cases.

Agents and tenants express their support for descriptors and actions using the following Key patterns:

- supported-<descriptor template name> indicates a support for the descriptor template as defined in its original specification. For example "supported-rfc5777classifier" is a Descriptor Template that conforms to the rfc5777-classifier (Figure 31) as defined in this document.

- supported-<action template name> indicates a support for the action template as defined in its original specification.

- "base-rule" is comprised of all base descriptors using an ‘or’ Descriptor-Match-Type and all Actions in no specific order.

- "base-template" is comprised of the base rule.

"base-template" can be used to determine supported Action and Descriptor Templates. It can also be used to support an open template where any specific Descriptors and Actions can be applied, however, depending upon the Order of Actions it is likely to produce undesirable results.

One use case is supported via reservation of specific DPN-Keys:

Requested policies are those that the Client would like to be assigned to a DPN within a Mobility-Context. The naming convention is similar to those used for DPN Assignment via an Agent.

- "Requested" is a Key that represents requested policies which have not been assigned to a specific DPN. No Role is assigned to the DPN.

- "Requested-<Role>" represents requested policies that have not been assigned to a DPN and can only be assigned to DPNs that fulfill the specified Role.
It is possible to have policies in the "Requested" DPN that do not appear in other entries which reflects the inability to successfully assign the policy.

5.1.2. Monitor Messages

An Agent may reject a registration if it or the DPN has insufficient resources.

An Agent or DPN MAY temporarily suspend monitoring if insufficient resources exist. In such a case the Agent MUST notify the Client.

When a monitor has a reporting configuration of SCHEDULED it is automatically de-registered after the last Notify occurs.

If a SCHEDULED or PERIODIC configuration is provided during registration with the time related value (time or period respectively) of 0 a Notify is sent and the monitor is immediately de-registered. This method should, when a Monitor has not been installed, result in an immediate Notify sufficient for the Client’s needs and lets the Agent realize the Client has no further need for the monitor to be registered.

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

If the Monitor configuration associated with a Notify can be deferred, then the Notify MAY be bundled with other messages back to the Agent even if this results in a delay of the Notify.

The Monitor messages use the following data:

Monitor-Key: Monitor Key.

Monitor: A Monitor configuration (see Section 4.9.7).

Send-Data: An indicator that specifies that the final value MUST be sent as a notification from the Agent.
5.1.2.1. Asynchronous Notification

A Monitor Report can be sent as part of de-registration, a trigger based upon a Monitor Configuration or a Probe. A Report is comprised of the Monitor Key the report applies to, the Trigger for the report, a timestamp of when the report’s associated event occurs and data, Report-Value, that is specific to the monitored value’s type.

Triggers include but are not limited to

- Subscribed Event occurred
- Low Threshold Crossed
- High Threshold Crossed
- Periodic Report
o Scheduled Report
o Probe
o Deregistration Final Value
o Monitoring Suspended
o Monitoring Resumed
o DPN Available
o DPN Unavailable

Multiple Reports are sent in a Notify message. Each Notify is comprised of unique Notification Identifier from the Agent and timestamp indicating when the notification was created.

```
+-[ Notify ]
  +-([Unsigned 32) Notification-Identifier:]
  +-[Timestamp:]
  +-[Report:] <List>
    +-[Monitor-Key:]
    +-[Report-Value]
```

Figure 20: Monitor Messages

5.2. Protocol Operation

Please note that JSON is used to represent the information in Figures in this section but any over the wire representation that accurately reflects the information model MAY be used.

5.2.1. DPN Selection

In order to assign a DPN to a Mobility Context, the Client or Agent requires topology information. The Service-Group provides information, e.g. function, role, protocol, features and configuration, to determine suitable DPN interfaces.

Consider a Client attempting to select DPN interfaces that are served by a single Agent. In this example interfaces are present with different protocols, settings and features as shown in the following figure.

"topology-information-model" : {
"dpn" : [ {
  "dpn-key" : "dpn1",
  "interface" : [ {
    "interface-key" : "ifc1",
    "role" : "lma",
    "protocol" : [ "pmip" ],
    "interface-configuration" : [ {
      "index" : 0,
      "setting" : [ "optionA" : "OFF" ]
    } ]
  },{
    "interface-key" : "ifc2",
    "role" : "lma",
    "protocol" : [ "pmip" ],
    "interface-configuration" : [ {
      "index" : 0,
      "setting" : [ "optionC" : "OFF" ]
    } ]
  },{
    "interface-key" : "ifc2-b",
    "role" : "mag",
    "protocol" : [ "pmip" ]
  } ] },
  "dpn-key" : "dpn2",
  "interface" : [ {
    "interface-key" : "ifc1",
    "role" : "mag",
    "protocol" : [ "pmip" ],
    "interface-configuration" : [ {
      "index" : 0,
      "settings" : [ "optionA" : "OFF", "optionB" : "ON" ]
    } ]
  } ] }
},

"service-group" : [ {
  "service-group-key" : "group1",
  "service-group-name" : "Anchors-OptionA-OFF",
  "role-key" : "lma",
  "protocol" : [ "pmip" ],
  "service-group-configuration" : [ {
    "index" : 0,
    "setting" : [ "optionA" : "OFF" ]
  } ]
},
  "dpn" : [ {
  "dpn-key" : "dpn2",
  "interface" : [ {
    "interface-key" : "ifc1",
    "role" : "mag",
    "protocol" : [ "pmip" ],
    "interface-configuration" : [ {
      "index" : 0,
      "setting" : [ "optionA" : "OFF" ]
    } ]
  } ] }
} ...
"referenced-interface" : [ { "interface-key" : "ifc1" } ] }
}
{"service-group-key" : "group2",
"service-group-name" : "Anchors",
"role-role" : "lma",
"protocol" : [ "pmip" ],
"dpn" : [ 
{ "dpn-key" : "dpn1",
"referenced-interface" : [ { "interface-key" : "ifc2" } ] }
}
{"service-group-key" : "group3",
"service-group-name" : "MAGs",
"role-role" : "mag",
"protocol" : [ "pmip" ],
"dpn" : [ 
{ "dpn-key" : "dpn2",
"referenced-interface" : [ { "interface-key" : "ifc1" } ] },
{ "dpn-key" : "dpn1",
"referenced-interface" : [ { "interface-key" : "ifc2-b" } ] }
]
}
]

NOTE - A Setting is, in this example, a list of string attributes in a Configuration.

Figure 21: Monitor Messages

Two DPNs are present. The first, dpn1, has 3 interfaces. Two support the LMA role and both have settings. The third supports the MAG function. The second DPN, dpn2, provides a single interface with the MAG function.

Three ServiceGroups are presented. The first provides the PMIP protocol and LMA role. It also has a setting, OptionA, that is OFF and only contains ifc1 from dpn1.

The second group is comprised of interfaces that support the PMIP protocol and LMA function. It only contains ifc2 from dpn1. An interface that has setting(s) or feature(s) that must appear in a ServiceGroup SHOULD NOT appear in ServiceGroups that do not have those setting(s) or feature(s) present. Thus, ifc1 of dpn1 should not be present in this second Service-Group.

A third group is comprised of interfaces that support the MAG function of the LMA protocol. It contains the MAG interfaces form both dpn1 and dpn2.
Given the task to find a LMA that supports the PMIP protocol the Client can determine that dpn1 is its only option and, depending on its requirement of OptionA, can appropriately determine which interface to select.

5.2.2. Policy Creation and Installation

A Policy must be installed upon an Agent in order to install policies on the selected dpn(s). This requires construction of the Action(s), Descriptor(s) and Rule(s) used by the Policy.

The CONFIGURE message permits editing all information elements except monitors. The following figure shows use of a CONFIGURE message to install policy information on the Agent.

```
MAG-C1 | MAG-C2 | LMA-C | Client | Agent | DPN
------+-------+-------+--------+-------+-------
|       |       |       |        |       |       |
| ------+-------+-------+--------+-------+-------|
|       |       |-------+--------+-------+-------|
|       |       |       |        |       |       |
|       |       |       |        |       |       |
```

In this example a Descriptor "all-traffic" Template and an Action, "drop", Template are both empty Templates. The "deny-all" Rule Template is comprised of the action and descriptor. The Rule is included in "policy1". The policy’s status is "Configured" as it is a complete policy ready for immediate use. The policy could be set as "Active" if the Client intends to use it upon immediate installation in a DPN.

Installation of the policy on dpn1 is shown in the following Figure. The Policy-Status is set to "Active" to make it immediately usable. Leaving the status as Configured would permit its installation on the DPN without an ability to use it in a Mobility Context. Such a use case is often referred to as policy pre-configuration.
5.2.3. Simple RPC Operation

A Client and Agent MUST identify themselves using the Client Identifier and Agent Identifier respectively to ensure that, for all transactions, a recipient of a FPC message can unambiguously identify the sender of the FPC message.

A Client MAY direct the Agent to enforce a rule in a particular DPN by including a DPN Key value in a Mobility Context. Otherwise the Agent selects a suitable DPN to enforce one or more portions of a Mobility Context and notifies the Client about the selected DPN(s) using DPN Identifier(s).

All messages sent from a Client to an Agent MUST be acknowledged by the Agent. The response must include all edit status as well as subsequent edits, which indicates the result of processing the message, as part of the Configure response. In case the processing of the message results in a failure, the Agent sets the global

Figure 23: Example Policy Installation (focus on FPC reference point)

This message uses an edit type of "create" to add the policy template directly to the installed DPN policy set.

```
"configure": {
  "client-id": 0,
  "operation-id": 1,
  "edit": [{
    "edit-id": 0,
    "edit-type": "create",
    "target": "/topology-information-model/dpn/dpn1/dpn-policy-configuration",
    "value": {
      "policy-template-key": "policy1",
      "policy-status": "active"
    }
  }
}
```

```
<--(2)- Response ------policy--->
{  
  "agent-id": "agent1",
  "operation-id": 1,
  "result-status": "ok"
}
```
status, Error-Type and Error-Tag accordingly and MAY clear the entity, e.g. Mobility-Context, which caused the failure, in the response.

If based upon Agent configuration or the processing of the request possibly taking a significant amount of time the Agent MAY respond with a Notify-Follows indication with optional Subsequent-Edit(s) containing the partially completed entity modifications. When a Notify-Follows indication is sent in a response, the Agent will, upon completion or failure of the operation, respond with an asynchronous Configuration-Result-Notification to the Client.

A Client MAY add a property to a Mobility-Context without providing all required details of the attribute’s value. In such case the Agent SHOULD determine the missing details and provide the completed property description, via Subsequent-Edit(s), back to the Client. If the processing will take too long or based upon Agent configuration, the Agent MAY respond with an OK for the Edit that indicates a Notify-Follows and also includes Subsequent-Edit(s) containing the partially completed entity edits.

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

Figure 24 illustrates an exemplary session life-cycle based on Proxy Mobile IPv6 registration via MAG Control-Plane function 1 (MAG-C1) and handover to MAG Control-Plane function 2 (MAG-C2). Edge DPN1 represents the Proxy CoA after attachment, whereas Edge DPN2 serves as Proxy CoA after handover. As exemplary architecture, the FPC Agent and the network control function are assumed to be co-located with the Anchor-DPN, e.g. a Router.

The Target of the second request uses the Mobility-Context by name. Alternatively, the Target could have included the DPN-Key and Policy-Key to further reduce the amount of information exchanged. Setting the Target’s value to the most specific node SHOULD be followed whenever practical.

+-------Router--------+
---(1)--Configure--------->

```json
"configure": {
  "client-id": 0,
  "operation-id": 3,
  "edit": {
    "edit-id": 0,
    "edit-type": "create",
    "target": "/mobility-context",
    "value": {
      "mobility-context-key": "ctxt1",
      "delegating-ip-prefix": [ <HNP> ],
      "dpn": [ {
        "dpn-key": "DPN1",
        "role": "lma",
        "service-data-flow": [ {
          "identifier": 0,
          "interface": [ "interface-key": "ifc1" ],
          "service-data-flow-policy-configuration": [{
            "policy-template-key": "dl-tunnel-with-gos",
            "policy-status": "active",
            "policy-configuration": [ {
              "index": 0,
              "qos-template": <QOS Settings...>,
              "index": 1,
              "tunnel": <DL tunnel info...>}
            ] } ] } ] } ]
    }
  }
}
```

---tun1 up-->
---tc qos-->
After reception of the Proxy Binding Update (PBU) at the LMA Control-Plane function (LMA-C), the LMA-C selects a suitable DPN, which serves as Data-Plane anchor to the mobile node’s (MN) traffic. The LMA-C adds a new logical Mobility-Context to the DPN to treat the MN’s traffic (1) and includes a Mobility-Context-Key (ctxt1) in the Configure command. The LMA-C identifies the selected Anchor DPN by including the associated DPN identifier.

The LMA-C adds policy template properties during the creation of the new Mobility-Context. One policy, "dl-tunnel-with-qos", is an example template that permits tunnel forwarding of traffic destined to the MN’s HNP, i.e. downlink traffic, with optional QoS parameters. Another policy, "ul-tunnel", provides a simple uplink anchor termination template where uplink tunnel information is provided.

The downlink tunnel information specifies the destination endpoint (Edge DPN1).

Upon reception of the Mobility-Context, the FPC Agent utilizes local configuration commands to create the tunnel (tun1) as well as the traffic control (tc) to enable QoS differentiation. After configuration has been completed, the Agent applies a new route to forward all traffic destined to the MN’s HNP specified as a property in the Mobility-Context and applied the configured tunnel interface (tun1).

During handover, the LMA-C receives an updating PBU from the handover target MAG-C2. The PBU refers to a new Data-Plane node (Edge DPN2) to represent the new tunnel endpoint in the downlink as required. The LMA-C sends a Configure message (3) to the Agent to modify the existing tunnel property of the existing Mobility-Context and to update the downlink tunnel endpoint from Edge DPN1 to Edge DPN2. Upon reception of the Configure message, the Agent applies updated tunnel property to the local configuration and responds to the Client (4).
When a teardown of the session occurs, MAG-C1 will send a PBU with a lifetime value of zero. The LMA-C sends a Configure message (1) to the Agent to modify the existing tunnel property of the existing

Figure 25: Single Agent with Deletion (focus on FPC reference point)
Mobility-Context to delete the tunnel information. Upon reception of the Configure message, the Agent removes the tunnel configuration and responds to the Client (2). Per [RFC5213], the PBA is sent back immediately after the PBA is received.

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

When a multi-DPN Agent is used the DPN list permits several DPNs to be provisioned in a single message for the single Mobility-Context.
"policy-configuration" : [
  "index" : 1,
  "tunnel" : <UL tunnel info...>]
],
"dpn-key" : "DPN2",
"role" : "mag",
"service-data-flow" : [
  "identifier" : 0,
  "interface" : [ "interface-key" : "ifc2" ],
  "service-data-flow-policy-configuration" :
    [ "policy-template-key" :
      "dl-tunnel-with-qos",
      "policy-status" : "active",
      "policy-configuration" : [
        { "index" : 0,
          "gos-template" : <QOS Settings...> },
        { "index" : 1,
          "tunnel" : <DL tunnel info...> },
        { "policy-template-key" : "ul-tunnel",
          "policy-status" : "active",
          "policy-configuration" : [
            { "index" : 1,
              "tunnel" : <UL tunnel info...> } ]
        } ] ] ]
],

<---(2)- Response ------<--route add>
{ "agent-id" : "agent1",
  "operation-id" : 0,
  "result-status" : "ok",
  "notify-follows" : "true",
}

<----------PBA------

+----+
| Edge|
| DPN2|
+----+

<---------------------- tun1 up ------------------
<---------------------- tc qos -------------------
<---------------------- route add ----------------

<(3) Configure-Result-Notification
{ "agent-id" : "agent1",

Figure 26: Exemplary Message Sequence for Multi-DPN Agent

Figure 26 shows how the first 2 messages in Figure 24 are supported when a multi-DPN Agent communicates with both Anchor DPN1 and Edge DPN2. In such a case, the FPC Client sends the downlink and uplink for both DPNs in the DPN Reference List of the same Mobility-Context. Message 1 shows the DPN Set with all entries. Each entry identifies the DPN.

The Agent responds with an OK and Notify-Follows indication while it simultaneously provisions both DPNs. Upon successful completion, the Agent responds to the Client with a Configuration-Result-Notification indicating the operation status.

5.2.4. Policy and Mobility on the Agent

A Client may build Policy and Topology using Configure messages.

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

Figure 27 illustrates the generic policy configuration model as used between a FPC Client and a FPC Agent.
As depicted in Figure 27, the DPN Settings represents the anchor of Rules through the Policy / Rule hierarchy. A Client and Agent use the identifier of the associated Policy to directly access the Rule and perform modifications of traffic Descriptors or Action references. Arriving packets are matched against traffic according to Rule precedence and Descriptors. If a Rule is applicable the packet is treated according to the ordered Action values.

A Client associates a Precedence value for the Rule’s Descriptors, to allow unambiguous traffic matching on the Data-Plane.

Figure 28 illustrates the generic context configuration model as used between a Client and an Agent.
Figure 28 represents a mobility session hierarchy. A Client and Agent directly assigns values such as downlink traffic descriptors, QoS information, etc. A Client and Agent use the context identifiers to access the descriptors, QoS information, etc. to perform modifications. From the viewpoint of packet processing, arriving packets are matched against traffic descriptors and processed according to QoS or other mobility profile related actions specified in the Mobility-Context’s and Service-Data-Flow’s properties. If present, a Policy could contain tunnel information to encapsulate and forward the packet.

A second Mobility-Context also references Mobility-Context-ID1 in the figure. Based upon the technology a property in a parent context (parent mobility-context-id reference) MAY be inherited by its descendants. This permits concise over the wire representation. When a Client deletes a parent Context all children are also deleted.

5.2.5. Monitor Example

The following example shows the installation of a DPN level monitor (1) to observe ifc1 status, a property that is either "up" or "down", and another monitor to watch for interface events. The interface experiences an outage which is reported to the Client via a Notify (3) message. At a later time a Probe (4) and corresponding Notify (5) is sent. Finally, the monitors are de-registered (6).
Note, specific event identifiers and types are out of scope.

```json
<---(1)--Configure-------->
"register-monitor": {
  "client-id": 0,
  "operation-id": 0,
  "monitor": [
    {
      "monitor-key": "ifc1-status",
      "target": "/dpn/dpn1/interface/ifc1/status"
    },
    {
      "monitor-key": "ifc1-events",
      "target": "/dpn/dpn1/interface/ifc1"
    }
  ],
  "configuration": {
    "target-event-configuration": [0, 1, 3, .. ]
  }
}

<---(2)-- Response -------
{
  "agent-id": "agent1",
  "operation-id": 0,
  "result-status": "ok"
}

[ ifc1 goes down which is reported as event type 3 ]
<---(3)-- NOTIFY --------
"notify": {
  "notification-id": 0,
  "timestamp": ..., 
  "report": [
    {"monitor-key": "ifc1-events",
     "trigger": "subscribed-event-occurred",
     "report-value": { 3 } }
  ]
}

---(4)-- Probe --------->
"probe": {
  "client-id": 0,
  "operation-id": 1,
  "monitor": {
    "monitor-key": "ifc1-status"
  }
}

<---(5)-- Response ------{
  "agent-id": "agent1",
"..."
Figure 29: Monitor Example (focus on FPC reference point)

6. Templates and Command Sets

Configuration templates are shown below.

6.1. Monitor Configuration Templates

A periodic configuration specifies a time interval (ms) for reporting.
A scheduled configuration specifies a time for reporting.

A threshold configuration MUST have at least one high or low threshold and MAY have both.

A Target-Events-Configuration is a list of Events that, when generated by the Target, results in a Monitor notification.

```
|   +-[Configuration]       
|     |     +-[Periodic-Configuration]       
|     |     |       -[(Unsigned32) Period:]       
|     +-[Configuration]       
|     |     +-[Schedule-Configuration]       
|     |     |       -[(Unsigned32) Schedule:]       
|     +-[Configuration]       
|     |     +-[Threshold-Configuration]  
|     |     |       -[(Unsigned32) Low]       
|     |     +-[Configuration]       
|     |     +-[Target-Events-Configuration]    
|     |     |       -[(Unsigned32) Event-Key:] <List> 
```

Figure 30: Monitor Configuration Templates

6.2. Descriptor Templates

A IP-Prefix-Template MUST have at least the To or From IP Prefix / Length populated. The IP Prefix specifies an Address and Length.

The PMIP Traffic Selector template is mapped according to [RFC6088].

The RFC 5777 Classifier is a structured version of common filter rules and follows the format specified in [RFC5777]. The Flow-Label, Flow-Label range and ECN-IP-Codepoint specified in [RFC7660] are added to the Descriptor as well.

```
|   +-[Configuration]       
|     |     +-[Configuration]       
|     |     |       -[(IP Prefix / Length) To-IP-Prefix]       
|     |     +-[Configuration]       
|     |     |       -[(IP Prefix / Length) From-IP-Prefix]       
|     +-[pmip-traffic-selector] 
```

Matsushima, et al. Expires December 20, 2018
+
  +[Enumerated - IPv4 or IPv6] ts-format
  +[ipsec-spi-range]
    +[ (ipsec-spi) start-spi: ]
    +[ (ipsec-spi) end-spi ]
  +[source-port-range]
    +[ (port-number) start-port: ]
    +[ (port-number) end-port ]
  +[destination-port-range]
    +[ (port-number) start-port: ]
    +[ (port-number) end-port ]
  +[source-address-range-v4]
    +[ (ipv4-address) start-address: ]
    +[ (ipv4-address) end-address ]
  +[destination-address-range-v4]
    +[ (ipv4-address) start-address: ]
    +[ (ipv4-address) end-address ]
  +[ds-range]
    +[ (dscp) start-ds: ]
    +[ (dscp) end-ds ]
  +[protocol-range]
    +[ (uint8) start-protocol: ]
    +[ (uint8) end-protocol ]
  +[source-address-range-v6]
    +[ (ipv6-address) start-address: ]
    +[ (ipv6-address) end-address ]
  +[destination-address-range-v6]
    +[ (ipv6-address) start-address: ]
    +[ (ipv6-address) end-address ]
  +[flow-label-range]
    +[ (ipv6-flow-label) start-flow-label ]
    +[ (ipv6-flow-label) end-flow-label ]
  +[traffic-class-range]
    +[ (dscp) start-traffic-class ]
    +[ (dscp) end-traffic-class ]
  +[next-header-range]
    +[ (uint8) start-next-header ]
    +[ (uint8) end-next-header ]
  ...
  +[rfc5777-classifier]
    +[Extensible: True]
    +[ (uint8) protocol ]
    +[Enumerated - In/Out/Both] Direction
    +[From-Spec] <List>
      +[ (ip-address) IP-Address] <List>
      +[ (ip-address) IP-Address-Range] <List>
      +[ (ip-address) IP-Address-Start]
      +[ (ip-address) IP-Address-End]
    +[ (ip-address) IP-Address-Mask] <List>
+-([ip-address] IP-Address:] 
  +-[Unsigned 32] IP-Bit-Mask-Width:
+-([mac-address] MAC-Address] <List>
  +-[MAC-Address-Mask] <List>
    +-[mac-address] MAC-Address:
      +-[MAC-Address-Mask-Pattern:]
+-([eui64-address] EUI64-Address] <List>
  +-[EUI64-Address-Mask] <List>
    +-[eui64-address] EUI64-Address:
      +-[eui64-address] EUI64-Address-Mask-Pattern:
+-([Integer 32] Port] <List>
  +-[Port-Range] <List>
    +-[Integer 32] Port-Start]
    +-[Integer 32] Port-End]
    +-[[Boolean] Negated]
  +-[[Boolean] Use-Assigned-Address]
+-[(To-Spec] <List> (0)
  +-[([ip-address] IP-Address] <List>
    +-[IP-Address-Range] <List>
      +-[([ip-address] IP-Address-Start]
      +-[([ip-address] IP-Address-End]
    +-[IP-Address-Mask] <List>
      +-[([ip-address] IP-Address:]
        +-[Unsigned 32] IP-Bit-Mask-Width:
    +-[([mac-address] MAC-Address] <List>
      +-[MAC-Address-Mask] <List>
        +-[mac-address] MAC-Address:
          +-[MAC-Address-Mask-Pattern:]
      +-[([eui64-address] EUI64-Address] <List>
        +-[EUI64-Address-Mask] <List>
          +-[([eui64-address] EUI64-Address:
            +-[([eui64-address] EUI64-Address-Mask-Pattern:
        +-[([Integer 32] Port] <List>
          +-[Port-Range] <List>
            +-[([Integer 32] Port-Start]
            +-[([Integer 32] Port-End]
          +-[([Boolean] Negated]
        +-[([Boolean] Use-Assigned-Address]
        +-[dscp] Diffserv-Code-Point] <List>
      +-[([Boolean] Fragmentation-Flag ~ False]
    +-[IP-Option] <List>
      +-[TCP-Option] <List>
        +-[TCP-Flags]
      +-[ICMP-Type] <List>
      +-[ETH-Option] <List>
      +-[ecn-ip-codepoint] <List>
        +-[((flowlabel) flow-label] <List>
      +-[flow-label-range] <List>
6.3. Tunnel Templates

The Network Service Header is specified in [RFC8300].

The MPLS SR Stack is specified in [I-D.ietf-spring-segment-routing-mpls].

The IPv6 SR Stack is specified in [I-D.ietf-6man-segment-routing-header].

A tunnel MUST have the local-address or remote-address (or both) populated.

For GRE, the gre-key MUST be present.

For GTP (GPRS Tunneling Protocol), the following attributes MAY be present

- local tunnel endpoint identifier (teid) - MUST be present if local-address is nonempty
- remote tunnel endpoint identifier (teid) - MUST be present if remote-address is nonempty
- sequence-numbers-on - Indicates that sequence numbers will be used

Tunnels can be used as Next Hop and Descriptor values.
Figure 32: Tunnel Templates

6.4. Action Templates

The following figure shows common next-hop (set next-hop) and tunnel templates for Actions.

Drop action has no values.

Rewrite uses a Descriptor to set the values of the packet. Exactly one Descriptor MUST be present. Only the Destination and Source port fields, if present, are used from the Descriptor.

Copy-Forward creates a copy of the packet and then forwards it in accordance to the next hop value.
6.5. Quality of Service Action Templates

PMIP QoS is specified in [RFC7222].

```
+[qos-template]
  +-[Extensible: True]
  +-[(dscp) trafficclass]
  +-[pmip-qos]
    +-[per-mn-agg-max-dl]
    +-[per-mn-agg-max-ul]
    +-[per-session-agg-max-dl]
      +-[max-rate:]
      +-[service-flag:]
      +-[exclude-flag:]
    +-[per-session-agg-max-ul]
      +-[max-rate:]
      +-[service-flag:]
      +-[exclude-flag:]
    +-[allocation-retention-priority]
      +-[priority-level:]
      +-[preemption-capability:]
    +-[agg-max-dl]
    +-[agg-max-ul]
    +-[gbr-dl]
    +-[gbr-ul]
```

Figure 34: QoS Templates
6.6. PMIP Command-Set

The following Command Set values are supported for IETF PMIP.

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

6.7. 3GPP Specific Templates and Command-Set

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

- APN-AMBR: Access Point Name Aggregate Maximum Bit Rate
- UE-AMBR: User Equipment Aggregate Maximum Bit Rate
- QCI: QoS Class Identifier
- EBI: EPS Bearer Identity
- LBI: Linked Bearer Identity
- IMSI: International Mobile Subscriber Identity
- TFT: Traffic Flow Template (TFT)

Generally, 3GPP QoS values should use the qos-template. Note: User Equipment Aggregate Maximum Bit Rate (UE-AMBR) maps to the per-mn-agg-max-dl and per-mn-agg-max-ul.
The following Command Set values are supported for 3GPP.
assign-ip - Assign the IP Address for the mobile session.

assign-fteid-ip - Assign the Fully Qualified TEID (F-TEID) LOCAL IP address.

assign-fteid-teid - Assign the Fully Qualified TEID (F-TEID) LOCAL TEID.

session - Assign values for the Session Level. When this involves 'assign-fteid-ip' and 'assign-fteid-teid', the values are part of the default bearer.

uplink - Command applies to uplink.

downlink - Command applies to downlink.

assign-dpn - Assign the Data-plane Node.

7. Implementation Status

Three FPC Agent implementations have been made to date. The first was based upon Version 03 of the draft and followed Model 1. The second follows Version 04 of the document. Both implementations were OpenDaylight plug-ins developed in Java by Sprint. Version 04 is now primarily enhanced by GS Labs. Version 03 was known as fpcagent and version 04’s implementation is simply referred to as ‘fpc’. A third has been developed on an ONOS Controller for use in MCORD projects.

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

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

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

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

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

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

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

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

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

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

Initial v04 performance results without code optimizations or tuning allow reliable provisioning of 1k FPC Mobility-Contexts processed per second on a 12 core server. This results in 2x the number of transactions on the southbound interface to a proprietary DPN API on the same machine.

fpc currently supports the following:

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

CONFIG and CONF_BUNDLE (all operations)

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

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

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

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

Figure 37: fpc pseudo code

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

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

8. Security Considerations

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

The YANG modules defined in this memo are designed to be accessed via the NETCONF [RFC6241] or RESTCONF [RFC8040] protocol. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242].

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

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

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

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

Mobility-Context provides runtime only information and manipulated by remote procedure calls. The unwanted deletion or removal of such information would deny users service or provide services to unauthorized parties.

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

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

International Mobile Subscriber Identity (IMSI) and bearer identifiers in the Context when using the FPC base model

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

Configure sends Mobility-Context information which can include information of a sensitive or vulnerable nature in some network environments as described above.

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

Notifications MUST be treated with same level of protection and scrutiny as the operations they correspond to. For example, a Configure-Result-Notification provides the same information that is sent as part of the input and output of the Configure RPC operation.

General usage of FPC MUST consider the following:

FPC Naming Section 4.5 permits arbitrary string values but a user MUST avoid placing sensitive or vulnerable information in those values.

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

9. IANA Considerations

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

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
prefix: fpc
reference: TBD1

name: ietf-dmm-pmip-qos
prefix: qos-pmip
reference: TBD2

name: ietf-dmm-traffic-selector-types
prefix: traffic-selectors
reference: TBD3

name: ietf-dmm-fpc-settingsext
prefix: fpcbase
reference: TBD4

name: ietf-diam-trafficclassifier
prefix: diamclassifier
reference: TBD5
10. Work Team Participants

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

11. References

11.1. Normative References

[I-D.ietf-6man-segment-routing-header]

[I-D.ietf-spring-segment-routing-mpls]


11.2. Informative References

[I-D.bertz-dime-policygroups]
Bertz, L. and M. Bales, "Diameter Policy Groups and Sets", draft-bertz-dime-policygroups-05 (work in progress), December 2017.

[I-D.ietf-dmm-deployment-models]


Appendix A. YANG Data Model for the FPC protocol

This section provides a type mapping for FPC structures in YANG. When being mapped to a specific information such as YANG the data type MAY change.

Keys for Actions, Descriptors, Rules, Policies, DPNs, Domains and Mobility-Contexts are specified as FPC-Identity which follows rules according to Section 4.5.

Action and Descriptor Templates are mapped as choices. This was done to ensure no duplication of Types and avoid use of identityref for typing.

Policy Expressions are provided as default values. NOTE that a static value CANNOT be supported in YANG.

Mapping of templates to YANG are performed as follows:

Value is defined as a choice statement for extensibility and therefore a type value is not necessary to discriminated types.

Generic attributes are distinguished by the "Settings" type and holds ANY value. It is an any data node under configurations.

The CONFIGURE and CONFIGURE-RESULT-NOTIFICATION use the yang-patch-status which is a container for edits. This was done to maximize YANG reuse.
In the configure rpc, operation-id is mapped to patch-id and in an edit the edit-type is mapped to operation.

The Result-Status attribute is mapped to the 'ok' (empty leaf) or errors structure.

The Policy-Status is mapped to entity-state to reduce YANG size.

Five modules are defined:

- **ietf-dmm-fpc (fpc)** - Defines the base model and messages for FPC that are meant to be static in FPC.
- **ietf-dmm-fpc-settingsext** - A FPC module that defines the information model elements that are likely to be extended in FPC.
- **ietf-pmip-qos (pmip-qos)** - Defines proxy mobile IPv6 QoS parameters per RFC 7222.
- **ietf-trafficselectors-types (traffic-selectors)** - Defines Traffic Selectors per [RFC6088].
- **ietf-diam-trafficclassifier (diamclassifier)** - Defines the Classifier per [RFC5777].

All modules defined in this specification make use of (import) ietf-inet-types as defined in [RFC6991].

ietf-dmm-fpc-settingsext and ietf-diam-trafficclassifier make use of (imports) ietf-yang-types as defined in [RFC6991].

ietf-dmm-fpc imports the restconf (ietf-restconf) [RFC8040] and yang patch (ietf-yang-patch) [RFC8072] modules.

ietf-pmip-qos and ietf-dmm-fpc-settings import the trafficselector from the ietf-traffic-selector-types module.

ietf-dmm-fpc-settings also imports the qosattribute (ietf-pmip-qos) and classifier (ietf-diam-trafficclassifier).

ietf-dmm-fpc-settingsext groups various settings, actions and descriptors and is used by the fpc module (ietf-dmm-fpc).

The following groupings are intended for reuse (import) by other modules.

- **qosoption (ietf-qos-pmip module)**
The YANG modules in this document conform to the Network Management
Datastore Architecture (NMDA) defined in [RFC8342].

DPNs conformant to NMDA MAY only have policies, installed policies,
topology, domains and mobility session information that has been
assigned to it in its intended and operational datastores. What is
housed in the operational datastore MAY be determined on a per DPN
basis and using the Entity-Status as a guideline based upon tradeoffs
described in Section 4.6.

ServiceGroups are not expected to appear in operational datastores of
DPNs as they remain in and are used by FPC Agents and Clients. They
MAY be operationally present in DNS when using the Dynamic Delegation
and Discovery System (DDDS) as defined in [RFC3958] or the
operational datastore of systems that provide equivalent
functionality.

A.1. FPC YANG Model

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

This module references [RFC6991], [RFC8040] and the fpc-settingsext
module defined in this document.

```yml
<CODE BEGINS> file "ietf-dmm-fpc@2018-05-17.yang"
module ietf-dmm-fpc {
  yang-version 1.1;
  prefix fpc;
```
import ietf-inet-types { prefix inet;  
revision-date 2013-07-15; }
import ietf-dmm-fpc-settingsex { prefix fpcbase;  
revision-date 2018-05-17; }
import ietf-diam-trafficclassifier { prefix rfc5777;  
revision-date 2018-05-17; }
import ietf-restconf { prefix rc;  
revision-date 2017-01-26; }
import ietf-yang-patch { prefix ypatch;  
revision-date 2017-02-22; }

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

contact  
"WG Web:  <http://tools.ietf.org/wg/netmod/>  
WG List:  <mailto:netmod@ietf.org>  
WG Chair: Dapeng Liu  
<mailto:maxpassion@gmail.com>  
WG Chair: Jouni Korhonen  
<mailto:jouni.nospam@gmail.com>  
Editor: Satoru Matsushima  
<mailto:satoru.matsushima@g.softbank.co.jp>  
Editor: Lyle Bertz  
<mailto:lylebe551144@gmail.com>";

description  
"This module contains YANG definition for  
Forwarding Policy Configuration Protocol (FPCP).  
Copyright (c) 2016 IETF Trust and the persons identified as the  
document authors. All rights reserved.  
This document is subject to BCP 78 and the IETF Trust’s Legal  
Provisions Relating to IETF Documents  
(http://trustee.ietf.org/license-info) in effect on the date of  
publishation of this document. Please review these documents  
carefully, as they describe your rights and restrictions with  
respect to this document. Code Components extracted from this  
document must include Simplified BSD License text as described  
in Section 4.e of the Trust Legal Provisions and are provided  
without warranty as described in the Simplified BSD License.";

revision 2018-05-17 {
description "Initial Revision.";
reference "draft-ietf-dmm-fpc-cpdp-10";
}

//General Structures
grouping templatedef {
  leaf extensible {
    type boolean;
    description "Indicates if the template is extensible";
  }
  leaf-list static-attributes {
    type string;
    description "Attribute (Name) whose value cannot change";
  }
  leaf-list mandatory-attributes {
    type string;
    description "Attribute (Name) of optional attributes that MUST be present in instances of this template.";
  }
  leaf entity-state {
    type enumeration {
      enum initial {
        description "Initial Configuration";
      }
      enum partially-configured {
        description "Partial Configuration";
      }
      enum configured {
        description "Configured";
      }
      enum active {
        description "Active";
      }
    }
    default initial;
    description "Entity State";
  }
  leaf version {
    type uint32;
    description "Template Version";
  }
  description "Template Definition";
}
typedef fpc-identity {
  type union {
    type uint32;
    type instance-identifier;
  }
}
type string;
}  
description "FPC Identity";
}  
grouping index {
  leaf index {
    type uint16;
    description "Index";
  }
  description "Index Value";
}

// Policy Structures

grouping descriptor-template-key {
  leaf descriptor-template-key {
    type fpc:fpc-identity;
    mandatory true;
    description "Descriptor Key";
  }
  description "Descriptor-Template Key";
}

grouping action-template-key {
  leaf action-template-key {
    type fpc:fpc-identity;
    mandatory true;
    description "Action Key";
  }
  description "Action-Template Key";
}

grouping rule-template-key {
  leaf rule-template-key {
    type fpc:fpc-identity;
    mandatory true;
    description "Rule Identifier";
  }
  description "Rule Key";
}

grouping policy-template-key {
  leaf policy-template-key {
    type fpc:fpc-identity;
    mandatory true;
    description "Rule Identifier";
  }
  description "Rule Key";
}

grouping fpc-setting-value {
  anydata setting;
}
description "FPC Setting Value";

// Configuration / Settings
grouping policy-configuration-choice {
    choice policy-configuration-value {
        case descriptor-value {
            uses fpcbase:fpc-descriptor-value;
            description "Descriptor Value";
        }
        case action-value {
            uses fpcbase:fpc-action-value;
            description "Action Value";
        }
        case setting-value {
            uses fpc:fpc-setting-value;
            description "Setting";
        }
        description "Policy Attributes";
    }
    description "Policy Configuration Value Choice";
}

grouping policy-configuration {
    list policy-configuration {
        key index;
        uses fpc:index;
        uses fpcbase:fpc-policy-configuration-choice;
        description "Policy Configuration";
    }
    description "Policy Configuration Value";
}

grouping ref-configuration {
    uses fpc:policy-template-key;
    uses fpcbase:fpc-policy-configuration;
    uses fpc:templatedef;
    description "Policy-Configuration Entry";
}

// FPC Policy

grouping policy-information-model {
    list action-template {
        key action-template-key;
        uses fpcbase:fpc-action-template-key;
        uses fpcbase:fpc-action-value;
        uses fpc:templatedef;
        description "Action Template";
    }
    list descriptor-template {
        key descriptor-template-key;
    }
}

uses fpc:descriptor-template-key;
uses fpcbase:fpc-descriptor-value;
uses fpc:templatedef;
description "Descriptor Template";
}
list rule-template {
  key rule-template-key;
  uses fpc:rule-template-key;
  leaf descriptor-match-type {
    type enumeration {
      enum or {
        value 0;
        description "OR logic";
      }
      enum and {
        value 1;
        description "AND logic";
      }
    }
    mandatory true;
    description "Type of Match (OR or AND) applied to the descriptor-configurations";
  }
}
list descriptor-configuration {
  key "descriptor-template-key";
  uses fpc:descriptor-template-key;
  leaf direction {
    type rfc5777:direction-type;
    description "Direction";
  }
}
list attribute-expression {
  key index;
  uses fpc:index;
  uses fpcbase:fpc-descriptor-value;
  description "Descriptor Attributes";
}
uses fpc:fpc-setting-value;
description "A set of Descriptor references";
}
list action-configuration {
  key "action-order";
  leaf action-order {
    type uint32;
    mandatory true;
    description "Action Execution Order";
  }
  uses fpc:action-template-key;
  list attribute-expression {

key index;
uses fpc:index;
uses fpcbase:fpc-action-value;
description "Action Attributes";
}
uses fpc:fpc-setting-value;
description "A set of Action references";
}
uses fpc:templatedef;
list rule-configuration {
  key index;
  uses fpc:index;
  uses fpc:policy-configuration-choice;
  description "Rule Configuration";
}
description "Rule Template";
}
list policy-template {
  key policy-template-key;
  uses fpc:policy-template-key;
  list rule-template {
    key "precedence";
    unique "rule-template-key";
    leaf precedence {
      type uint32;
      mandatory true;
      description "Rule Precedence";
    }
  }
  uses fpc:rule-template-key;
  description "Rule Entry";
}
uses fpc:templatedef;
uses fpc:policy-configuration;
description "Policy Template";
}
description "FPC Policy Structures";
}

// Topology Information Model
identity role { 
  description "Role";
}

grouping dpn-key { 
  leaf dpn-key {
    type fpc:fpc-identity;
    description "DPN Key";
  }
  description "DPN Key";
}
grouping role-key {
  leaf role-key {
    type identityref {
      base "fpc:role";
    }
    mandatory true;
    description "Access Technology Role";
  }
  description "Access Technology Role key";
}

grouping interface-key {
  leaf interface-key {
    type fpc:fpc-identity;
    mandatory true;
    description "interface identifier";
  }
  description "Interface Identifier key";
}

identity interface-protocols {
  description "Protocol supported by the interface";
}

identity features {
  description "Protocol features";
}

// Mobility Context

grouping mobility-context {
  leaf mobility-context-key {
    type fpc:fpc-identity;
    mandatory true;
    description "Mobility Context Key";
  }
  leaf-list delegating-ip-prefix {
    type inet:ip-prefix;
    description "IP Prefix";
  }
  leaf parent-context {
    type fpc:fpc-identity;
    description "Parent Mobility Context";
  }
  leaf-list child-context {
    type fpc:fpc-identity;
    description "Child Mobility Context";
  }
  container mobile-node {
    leaf-list ip-address {
      type inet:ip-address;
    }
  }
}
description "IP Address";
}
leaf imsi {
  type fpcbase:imsi-type;
  description "IMSI";
}
list mn-policy-configuration {
  key policy-template-key;
  uses fpc:ref-configuration;
  description "MN Policy Configuration";
  description "Mobile Node";
}
container domain {
  leaf domain-key {
    type fpc:fpc-identity;
    description "Domain Key";
  }
  list domain-policy-settings {
    key policy-template-key;
    uses fpc:ref-configuration;
    description "MN Policy Configuration";
  }
  description "Domain";
}
list dpn {
  key dpn-key;
  uses fpc:dpn-key;
  list dpn-policy-configuration {
    key policy-template-key;
    uses fpc:ref-configuration;
    description "DPN Policy Configuration";
  }
  leaf role {
    type identityref {
      base "fpc:role";
    }
    description "Role";
  }
  list service-data-flow {
    key identifier;
    leaf identifier {
      type uint32;
      description "Generic Identifier";
    }
    leaf service-group-key {
      type fpc:fpc-identity;
      description "Service Group Key";
    }
list interface {
    key interface-key;
    uses fpc:interface-key;
    description "interface assigned";
}

list service-data-flow-policy-configuration {
    key policy-template-key;
    uses fpc:ref-configuration;
    description "Flow Policy Configuration";
}

description "Service Dataflow";

description "DPN";

description "Mobility Context";

// Events, Probes & Notifications
identity event-type {
    description "Base Event Type";
}
typedef event-type-id {
    type uint32;
    description "Event ID Type";
}
grouping monitor-key {
    leaf monitor-key {
        type fpc:fpc-identity;
        mandatory true;
        description "Monitor Key";
    }
    description "Monitor Id";
}
grouping monitor-config {
    uses fpc:templatedef;
    uses fpc:monitor-key;
    leaf target {
        type string;
        description "target";
    }
    leaf deferrable {
        type boolean;
        description "Indicates reports related to this config can be delayed.";
    }
    choice configuration {
        mandatory true;
leaf period {
    type uint32;
    description "Period";
}

case threshold-config {
    leaf low {
        type uint32;
        description "low threshold";
    }
    leaf hi {
        type uint32;
        description "high threshold";
    }
    description "Threshold Config Case";
}

leaf schedule {
    type uint32;
    description "Reporting Time";
}

leaf-list event-identities {
    type identityref {
        base "fpc:event-type";
    }
    description "Event Identities";
}

leaf-list event-ids {
    type uint32;
    description "Event IDs";
}

description "Event Config Value";
description "Monitor Configuration";

// Top Level Structures
list tenant {
    key "tenant-key";
    leaf tenant-key {
        type fpc:fpc-identity;
        description "Tenant Key";
    }
}

container topology-information-model {
    config false;
    list service-group {
        key "service-group-key role-key";
        leaf service-group-key {
            type fpc:fpc-identity;
            mandatory true;
        }
    }
}
description "Service Group Key";
}
leaf service-group-name {
  type string;
  description "Service Group Name";
}
uses fpc:role-key;
leaf role-name {
  type string;
  mandatory true;
  description "Role Name";
}
leaf-list protocol {
  type identityref {
    base "interface-protocols";
  }
  min-elements 1;
  description "Supported protocols";
}
leaf-list feature {
  type identityref {
    base "interface-protocols";
  }
  description "Supported features";
}
list service-group-configuration {
  key index;
  uses fpc:index;
  uses fpc:policy-configuration-choice;
  description "Settings";
}
list dpn {
  key dpn-key;
  uses fpc:dpn-key;
  min-elements 1;
  list referenced-interface {
    key interface-key;
    uses fpc:interface-key;
    leaf-list peer-service-group-key {
      type fpc:fpc-identity;
      description "Peer Service Group";
    }
    description "Referenced Interface";
  }
  description "DPN";
}
description "Service Group";
list dpn {
    key dpn-key;
    uses fpc:dpn-key;
    leaf dpn-name {
        type string;
        description "DPN name";
    }
    leaf dpn-resource-mapping-reference {
        type string;
        description "Reference to underlying DPN resource(s)";
    }
    leaf domain-key {
        type fpc:fpc-identity;
        description "Domains";
    }
    leaf-list service-group-key {
        type fpc:fpc-identity;
        description "Service Group";
    }
} list interface {
    key "interface-key";
    uses fpc:interface-key;
    leaf interface-name {
        type string;
        description "Service Endpoint Interface Name";
    }
    leaf role {
        type identityref {
            base "fpc:role";
        }
        description "Roles supported";
    }
    leaf-list protocol {
        type identityref {
            base "interface-protocols";
        }
        description "Supported protocols";
    }
} list interface-configuration {
    key index;
    uses fpc:index;
    uses fpc:policy-configuration-choice;
    description "Interface settings";
} description "DPN interfaces";
} list dpn-policy-configuration {
    key policy-template-key;
uses fpc:ref-configuration;
description "DPN Policy Configuration";
}
description "Set of DPNs";
}
list domain {
  key domain-key;
  leaf domain-key {
    type fpc:fpc-identity;
    mandatory true;
    description "Domain Key";
  }
  leaf domain-name {
    type string;
    description "Domain displayname";
  }
  list domain-policy-configuration {
    key policy-template-key;
    uses fpc:ref-configuration;
    description "Domain Configuration";
  }
  description "List of Domains";
}
container dpn-checkpoint {
  uses fpc:basename-info;
  description "DPN Checkpoint information";
}
container service-group-checkpoint {
  uses fpc:basename-info;
  description "Service Group Checkpoint information";
}
container domain-checkpoint {
  uses fpc:basename-info;
  description "Domain Checkpoint information";
}
description "FPC Topology grouping";
}
container policy-information-model {
  config false;
  uses fpc:policy-information-model;
  uses fpc:basename-info;
  description "Policy";
}
list mobility-context {
  key "mobility-context-key";
  config false;
  uses fpc:mobility-context;
  description "Mobility Context";
list monitor {
  key monitor-key;
  config false;
  uses fpc:monitor-config;
  description "Monitor";
}

description "Tenant";

typedef agent-identifier {
  type fpc:fpc-identity;
  description "Agent Identifier";
}
typedef client-identifier {
  type fpc:fpc-identity;
  description "Client Identifier";
}
grouping basename-info {
  leaf basename {
    type fpc:fpc-identity;
    description "Rules Basename";
  }
  leaf base-checkpoint {
    type string;
    description "Checkpoint";
  }
  description "Basename Information";
}

// RPCs

grouping client-id {
  leaf client-id {
    type fpc:client-identifier;
    mandatory true;
    description "Client Id";
  }
  description "Client Identifier";
}
grouping execution-delay {
  leaf execution-delay {
    type uint32;
    description "Execution Delay (ms)";
  }
  description "Execution Delay";
}
typedef ref-scope {
  type enumeration {
enum none {
    value 0;
    description "no references";
}
enum op {
    value 1;
    description "All references are intra-operation";
}
enum bundle {
    value 2;
    description "All references in exist in bundle";
}
enum storage {
    value 3;
    description "One or more references exist in storage.";
}
enum unknown {
    value 4;
    description "The location of the references are unknown.";
}

description "Search scope for references in the operation.";
}
rpc configure {
    description "Configure RPC";
    input {
        uses client-id;
        uses execution-delay;
        uses ypatch:yang-patch;
    }
    output {
        uses ypatch:yang-patch-status;
    }
}

augment "/configure/input/ yang-patch/edit" {
    leaf reference-scope {
        type fpc:ref-scope;
        description "Reference Scope";
    }
    uses fpcbase:instructions;
    description "yang-patch edit augments for configure rpc";
}

grouping subsequent-edits {
    list subsequent-edit {
        key edit-id;
        ordered-by user;

        description "Edit list";
    }
}
leaf edit-id {
    type string;
    description "Arbitrary string index for the edit."
}

leaf operation {
    type enumeration {
        enum create {
            description "Create"
        }
        enum delete {
            description "Delete"
        }
        enum insert {
            description "Insert"
        }
        enum merge {
            description "Merge"
        }
        enum move {
            description "Move"
        }
        enum replace {
            description "Replace"
        }
        enum remove {
            description "Delete the target node if it currently exists."
        }
    }
    mandatory true;
    description "The datastore operation requested"
}

leaf target {
    type ypatch:target-resource-offset;
    mandatory true;
    description "Identifies the target data node"
}

leaf point {
    when "(../operation = 'insert' or ../operation = 'move')" + "and (../where = 'before' or ../where = 'after')" {
        description "This leaf only applies for 'insert' or 'move' operations, before or after an existing entry."
    }
}
type ypatch:target-resource-offset;
  description
    "The absolute URL path for the data node";
}

leaf where {
  when "../operation = 'insert' or ../operation = 'move'" {
    description
      "This leaf only applies for 'insert' or 'move'
       operations.";
  }
  type enumeration {
    enum before {
      description
        "Insert or move a data node before.";
    }
    enum after {
      description
        "Insert or move a data node after.";
    }
    enum first {
      description
        "Insert or move a data node so it becomes ordered
         as the first entry.";
    }
    enum last {
      description
        "Insert or move a data node so it becomes ordered
         as the last entry.";
    }
  }
  default last;
  description
    "Identifies where a data resource will be inserted
     or moved.";
}

anydata value {
  when "../operation = 'create'
    + "or ../operation = 'merge'
    + "or ../operation = 'replace'
    + "or ../operation = 'insert'" {
    description
      "The anydata 'value' is only used for 'create',
       'merge', 'replace', and 'insert' operations.";
  }
  description

augment
"/configure/output/yang-patch-status/edit-status/edit/
+ "edit-status-choice/ok" {
leaf notify-follows {
    type boolean;
    description "Notify Follows Indication";
}
uses fpc:subsequent-edits;
description "Configure output augments";
}
grouping op-header {
uses client-id;
uses execution-delay;
leaf operation-id {
    type uint64;
    mandatory true;
    description "Operation Identifier";
}
description "Common Operation header";
}
grouping monitor-response {
leaf operation-id {
    type uint64;
    mandatory true;
    description "Operation Identifier";
}
choice edit-status-choice {
    description
    "A choice between different types of status responses for each 'edit' entry.";
leaf ok {
    type empty;
    description
    "This 'edit' entry was invoked without any errors detected by the server associated
    with this edit.";
}
case errors {
    uses rc:errors;
    description
    "The server detected errors associated with the edit identified by the same 'edit-id' value.";
}


// Common RPCs
rpc register_monitor {
  description "Used to register monitoring of parameters/events";
  input {
    uses fpc:op-header;
    list monitor {
      key monitor-key;
      uses fpc:monitor-config;
      description "Monitor Configuration"
    }
  }
  output {
    uses fpc:monitor-response;
  }
}

rpc deregister_monitor {
  description "Used to de-register monitoring of parameters/events";
  input {
    uses fpc:op-header;
    list monitor {
      key monitor-key;
      uses fpc:monitor-key;
      min-elements 1;
      leaf send_data {
        type boolean;
        description "Indicates if NOTIFY with final data is desired upon deregistration"
      }
    }
    description "Monitor Identifier"
  }
  output {
    uses fpc:monitor-response;
  }
}

rpc probe {
  description "Probe the status of a registered monitor";
  input {
    uses fpc:op-header;
    list monitor {
      key monitor-key;
      uses fpc:monitor-key;
      min-elements 1;
    }
  }
}

description "Monitor";
}
}
output {
  uses fpc:monitor-response;
}
}

// Notification Messages & Structures
notification config-result-notification {
  uses ypatch:yang-patch-status;
  description "Configuration Result Notification";
}
augment "/config-result-notification" {
  uses fpc:subsequent-edits;
  description "config-result-notification augment";
}

identity notification-cause {
  description "Notification Cause";
}
identity subscribed-event-occurred {
  base "notification-cause";
  description "Subscribed Event Occurrence";
}
identity low-threshold-crossed {
  base "notification-cause";
  description "Subscribed Event Occurrence";
}
identity high-threshold-crossed {
  base "notification-cause";
  description "Subscribed Event Occurrence";
}
identity periodic-report {
  base "notification-cause";
  description "Periodic Report";
}
identity scheduled-report {
  base "notification-cause";
  description "Scheduled Report";
}
identity probe {
  base "notification-cause";
  description "Probe";
}
identity deregistration-final-value {
  base "notification-cause";
  description "Probe";
}
identity monitoring-suspension {
    base "notification-cause";
    description "Indicates monitoring suspension";
}

identity monitoring-resumption {
    base "notification-cause";
    description "Indicates that monitoring has resumed";
}

identity dpn-available {
    base "notification-cause";
    description "DPN Candidate Available";
}

identity dpn-unavailable {
    base "notification-cause";
    description "DPN Unavailable";
}

notification notify {
    leaf notification-id {
        type uint32;
        description "Notification Identifier";
    }
    leaf timestamp {
        type uint32;
        description "timestamp";
    }
    list report {
        key monitor-key;
        uses fpc:monitor-key;
        min-elements 1;
        leaf trigger {
            type identityref {
                base "notification-cause";
            }
            description "Notification Cause";
        }
        choice value {
            case dpn-candidate-available {
                leaf node-id {
                    type inet:uri;
                    description "Topology URI";
                }
                list supported-interface-list {
                    key role-key;
                    uses fpc:role-key;
                    description "Support Interfaces";
                }
                description "DPN Candidate Information";
            }
        }
    }
}
A.2. FPC YANG Settings and Extensions Model

This module defines the base data elements in FPC that are likely to be extended.

This module references [RFC6991], ietf-trafficselector-types and ietf-pmip-qos modules.

<CODE BEGINS> file "ietf-dmm-fpc-settingsext@2018-05-17.yang"
module ietf-dmm-fpc-settingsext {
    yang-version 1.1;
    prefix fpcbase;

    import ietf-inet-types { prefix inet;
        revision-date 2013-07-15; }
    import ietf-trafficselector-types { prefix traffic-selectors;
        revision-date 2018-05-17; }
    import ietf-yang-types { prefix ytypes;
        revision-date 2013-07-15; }
    import ietf-pmip-qos { prefix pmipqos;
        revision-date 2018-05-17; }
    import ietf-diam-trafficclassifier { prefix rfc5777;
        revision-date 2018-05-17; }

    organization "IETF Distributed Mobility Management (DMM) Working Group";
}
description
"This module contains YANG definition for Forwarding Policy Configuration Protocol (FPCP).

It contains Settings definitions as well as Descriptor and Action extensions.

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

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

revision 2018-05-17 {
    description "Initial Revision.";
    reference "draft-ietf-dmm-fpc-cpdp-10";
}

//Tunnel Information
identity tunnel-type {
    description "Tunnel Type";
}
identity grev1 {
    base "fpcbase:tunnel-type";
    description "GRE v1";
}
identity grev2 {
  base "fpcbase:tunnel-type";
  description "GRE v2";
}

identity ipinip {
  base "fpcbase:tunnel-type";
  description "IP in IP";
}

identity gtpv1 {
  base "fpcbase:tunnel-type";
  description "GTP version 1 Tunnel";
}

identity gtpv2 {
  base "fpcbase:tunnel-type";
  description "GTP version 2 Tunnel";
}

grouping tunnel-value {
  container tunnel-info {
    leaf tunnel-local-address {
      type inet:ip-address;
      description "local tunnel address";
    }
    leaf tunnel-remote-address {
      type inet:ip-address;
      description "remote tunnel address";
    }
    leaf mtu-size {
      type uint32;
      description "MTU size";
    }
    leaf tunnel {
      type identityref {
        base "fpcbase:tunnel-type";
      }
      description "tunnel type";
    }
    leaf payload-type {
      type enumeration {
        enum ipv4 {
          value 0;
          description "IPv4";
        }
        enum ipv6 {
          value 1;
          description "IPv6";
        }
      }
    }
  }
}
enum dual {
    value 2;
    description "IPv4 and IPv6";
}

leaf gre-key {
    type uint32;
    description "GRE_KEY";
}

container gtp-tunnel-info {
    leaf local-tunnel-identifier {
        type uint32;
        description "Tunnel Endpoint IDentifier (TEID)";
    }
    leaf remote-tunnel-identifier {
        type uint32;
        description "Tunnel Endpoint IDentifier (TEID)";
    }
    leaf sequence-numbers-enabled {
        type boolean;
        description "Sequence No. Enabled";
    }
    description "GTP Tunnel Information";
}

leaf ebi {
    type fpcbase:ebi-type;
    description "EPS Bearier Identifier";
}

leaf lbi {
    type fpcbase:ebi-type;
    description "Linked Bearier Identifier";
}

description "Tunnel Information";

description "Tunnel Value";

// DESCRIPTOR DEFINITIONS

// From 3GPP TS 24.008 version 13.5.0 Release 13
typedef packet-filter-direction {
    type enumeration {
        enum preRel7Tft {
            value 0;
            description "Pre-Release 7 TFT";
        }
    }
}

Matsushima, et al. Expires December 20, 2018 [Page 100]
enum uplink {
    value 1;
    description "uplink";
}

enum downlink {
    value 2;
    description "downlink";
}

enum bidirectional {
    value 3;
    description "bi-directional";
}

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 fpccbase: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 fpccbase:component-type-id;
            description "Component Type";
        }

        choice value {
            leaf ipv4-local {
                type inet:ipv4-address;
            }
        }
    }
}

description "IPv4 Local Address";
}
leaf ipv6-prefix-local {
    type inet:ipv6-prefix;
    description "IPv6 Local Prefix";
}
leaf ipv4-ipv6-remote {
    type inet:ip-address;
    description "Ipv4 Ipv6 remote address";
}
leaf ipv6-prefix-remote {
    type inet:ipv6-prefix;
    description "IPv6 Remote Prefix";
}
leaf next-header {
    type uint8;
    description "Next Header";
}
leaf local-port {
    type inet:port-number;
    description "Local Port";
}

case local-port-range {
    leaf local-port-lo {
        type inet:port-number;
        description "Local Port Min Value";
    }
    leaf local-port-hi {
        type inet:port-number;
        description "Local Port Max Value";
    }
}
leaf remote-port {
    type inet:port-number;
    description "Remote Port";
}

case remote-port-range {
    leaf remote-port-lo {
        type inet:port-number;
        description "Remote Port Min Value";
    }
    leaf remote-port-hi {
        type inet:port-number;
        description "Remote Port Max Value";
    }
}
leaf ipsec-index {
    type traffic-selectors:ipsec-spi;
description "IPSec Index";
} leaf traffic-class {
    type inet:dscp;
    description "Traffic Class";
} case traffic-class-range {
    leaf traffic-class-lo {
        type inet:dscp;
        description "Traffic Class Min Value";
    }
    leaf traffic-class-hi {
        type inet:dscp;
        description "Traffic Class Max Value";
    }
} leaf-list flow-label {
    type inet:ipv6-flow-label;
    description "Flow Label";
} description "Component Value";
}

description "Packet Filter";
}

grouping prefix-descriptor {
    leaf destination-ip {
        type inet:ip-prefix;
        description "Rule of destination IP";
    }
    leaf source-ip {
        type inet:ip-prefix;
        description "Rule of source IP";
    }
    description "Traffic descriptor based upon source/destination as IP prefixes";
}

grouping fpc-descriptor-value {
    choice descriptor-value {
        mandatory true;
        leaf all-traffic {
            type empty;
            description "admit any";
        }
        leaf no-traffic {
            type empty;
        }
    }
description "deny any";
}
case prefix-descriptor {
    uses fpcbase:prefix-descriptor;
    description "IP Prefix descriptor";
}
case pmip-selector {
    uses traffic-selectors:traffic-selector;
    description "PMIP Selector";
}
container rfc5777-classifier-template {
    uses rfc5777:classifier;
    description "RFC 5777 Classifier";
}
container packet-filter {
    uses fpcbase:packet-filter;
    description "Packet Filter";
}
case tunnel-info {
    uses fpcbase:tunnel-value;
    description "Tunnel Descriptor (only considers source info)";
}
description "Descriptor Value";
}
description "FPC Descriptor Values";
}

// Next Hop Structures
typedef fpc-service-path-id {
    type uint32 {
        range "0..33554431";
    }
    description "SERVICE_PATH_ID";
}
typedef fpc-mpls-label {
    type uint32 {
        range "0..1048575";
    }
    description "MPLS label";
}
typedef segment-id {
    type string {
        length "16";
    }
    description "SR Segment Identifier";
}
grouping fpc-nexthop {
choice next-hop-value {
  leaf ip-address {
    type inet:ip-address;
    description "IP Value";
  }
  leaf mac-address {
    type ytypes:mac-address;
    description "MAC Address Value";
  }
  leaf service-path {
    type fpcbase:fpc-service-path-id;
    description "Service Path Value";
  }
  leaf mpls-path {
    type fpcbase:fpc-mpls-label;
    description "MPLS Value";
  }
  leaf nsh {
    type string {
      length "16";
    }
    description "Network Service Header";
  }
  leaf interface {
    type uint16;
    description "If (interface) Value";
  }
  leaf segment-identifier {
    type fpcbase:segment-id;
    description "Segment Id";
  }
  leaf-list mpls-label-stack {
    type fpcbase:fpc-mpls-label;
    description "MPLS Stack";
  }
  leaf-list mpls-sr-stack {
    type fpcbase:fpc-mpls-label;
    description "MPLS SR Stack";
  }
  leaf-list srv6-stack {
    type fpcbase:segment-id;
    description "Segment Id";
  }
  case tunnel-info {
    uses fpcbase:tunnel-value;
    description "Tunnel Descriptor (only considers source info)";
  }
}
typedef pmip-commandset {
    type bits {
        bit assign-ip {
            position 0;
            description "Assign IP";
        }
        bit assign-dpn {
            position 1;
            description "Assign DPN";
        }
        bit session {
            position 2;
            description "Session Level";
        }
        bit uplink {
            position 3;
            description "Uplink";
        }
        bit downlink {
            position 4;
            description "Downlink";
        }
    }
    description "PMIP Instructions";
}

typedef fpc-qos-class-identifier {
    type uint8 {
        range "1..9";
    }
    description "QoS Class Identifier (QCI)";
}
typedef ebi-type {
    type uint8 {
        range "0..15";
    }
    description "EUTRAN Bearere Identifier (EBI) Type";
}
typedef imsi-type {
  type uint64;
  description
    "International Mobile Subscriber Identity (IMSI)
     Value Type";
}

// Instructions
typedef threegpp-instr {
  type bits {
    bit assign-ip {
      position 0;
      description "Assign IP Address/Prefix";
    }
    bit assign-fteid-ip {
      position 1;
      description "Assign FTEID-IP";
    }
    bit assign-fteid-teid {
      position 2;
      description "Assign FTEID-TEID";
    }
    bit session {
      position 3;
      description "Commands apply to the Session Level";
    }
    bit uplink {
      position 4;
      description "Commands apply to the Uplink";
    }
    bit downlink {
      position 5;
      description "Commands apply to the Downlink";
    }
    bit assign-dpn {
      position 6;
      description "Assign DPN";
    }
  }
  description "Instruction Set for 3GPP R11";
}

////////////////////////////////////////////////////
// ACTION VALUE AUGMENTS
grouping fpc-action-value {
  choice action-value {
    mandatory true;
    leaf drop {
      type empty;
    }
  }
}
description "Drop Traffic";

} container rewrite {
  choice rewrite-value {
    case prefix-descriptor {
      uses fpcbase:prefix-descriptor;
      description "IP Prefix descriptor";
    }
    case pmip-selector {
      uses traffic-selectors:traffic-selector;
      description "PMIP Selector";
    }
    container rfc5777-classifier-template {
      uses rfc5777:classifier;
      description "RFC 5777 Classifier";
    }
    description "Rewrite Choice";
  }
  description "Rewrite/NAT value";
}

container copy-forward-nexthop {
  uses fpcbase:fpc-nexthop;
  description "Copy Forward Value";
}

container nexthop {
  uses fpcbase:fpc-nexthop;
  description "NextHop Value";
}

case qos {
  leaf trafficclass {
    type inet:dscp;
    description "Traffic Class";
  }
  uses pmipqos:qosattribute;
  leaf qci {
    type fpcbase:fpc-qos-class-identifier;
    description "QCI";
  }
  leaf ue-agg-max-bitrate {
    type uint32;
    description "UE Aggregate Max Bitrate";
  }
  leaf apn-ambr {
    type uint32;
    description "Access Point Name Aggregate Max Bit Rate";
  }
  description "QoS Attributes";
A.3. PMIP QoS Model

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

This module references [RFC6991].
revision-date 2018-05-17;

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

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

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

WG Chair: Sri Gundavelli <mailto:sgundave@cisco.com>

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

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

description
"This module contains a collection of YANG definitions for quality of service parameters used in Proxy Mobile IPv6.

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

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

revision 2018-05-17 {
    description "Initial Revision.";
    reference "RFC 6088: Traffic Selectors for Flow Bindings";
}

// Type Definitions

// QoS Option Field Type Definitions
typedef sr-id {
    type uint8;

The Natural Text is as follows:

```
revision-date 2018-05-17;

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

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

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

WG Chair: Sri Gundavelli <mailto:sgundave@cisco.com>

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

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

description
"This module contains a collection of YANG definitions for quality of service parameters used in Proxy Mobile IPv6.

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

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

revision 2018-05-17 {
    description "Initial Revision.";
    reference "RFC 6088: Traffic Selectors for Flow Bindings";
}

// Type Definitions

// QoS Option Field Type Definitions
typedef sr-id {
    type uint8;

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";
        }
    }
}
typedef Per-MN-Agg-Max-DL-Bit-Rate-Value {
  type uint32;
  description
  "The aggregate maximum downlink bit rate that is
  requested/allocated for all the mobile node’s IP flows.
  The measurement units are bits per second.";
}

typedef Per-MN-Agg-Max-UL-Bit-Rate-Value {
  type uint32;
  description
  "The aggregate maximum uplink bit rate that is
  requested/allocated for the mobile node’s IP flows. The
  measurement units are bits per second.";
}

// Generic Structure for the uplink and downlink
// Generic Structure for the uplink and downlink
// Generic Structure for the uplink and downlink
grouping Per-Session-Agg-Max-Bit-Rate-Value {
  leaf max-rate {
    type uint32;
    mandatory true;
    description
    "The aggregate maximum bit rate that is requested/allocated
    for all the IP flows associated with that mobility session.
    The measurement units are bits per second.";
  }
  leaf service-flag {
    type boolean;
    mandatory true;
    description
    "This flag is used for extending the scope of the
    target flows for Per-Session-Agg-Max-UL/DL-Bit-Rate
    from(UL)/to(DL) the mobile node’s other mobility sessions
    sharing the same Service Identifier.";
    reference
    "RFC 5149 - Service Selection mobility option";
  }
  leaf exclude-flag {
    type boolean;
    mandatory true;
  }
}
description
"This flag is used to request that the uplink/downlink
flows for which the network is providing
Guaranteed-Bit-Rate service be excluded from the
target IP flows for which
Per-Session-Agg-Max-UL/DL-Bit-Rate is measured.";
}
description "Per-Session-Agg-Max-Bit-Rate Value";

grouping Allocation-Retention-Priority-Value {
  leaf priority-level {
    type uint8 {
      range "0..15";
    }
    mandatory true;
description
"This is a 4-bit unsigned integer value. It is used to decide
whether a mobility session establishment or modification
request can be accepted; this is typically used for
admission control of Guaranteed Bit Rate traffic in case of
resource limitations.";
  }
  leaf preemption-capability {
    type enumeration {
      enum enabled {
        value 0;
        description "enabled";
      }
      enum disabled {
        value 1;
        description "disabled";
      }
      enum reserved1 {
        value 2;
        description "reserved1";
      }
      enum reserved2 {
        value 3;
        description "reserved2";
      }
    }
    mandatory true;
description
"This is a 2-bit unsigned integer value. It defines whether a
service data flow can get resources that were already
assigned to another service data flow with a lower priority
level.";
  }
}

leaf preemption-vulnerability {
  type enumeration {
    enum enabled {
      value 0;
      description "enabled";
    }
    enum disabled {
      value 1;
      description "disabled";
    }
    enum reserved1 {
      value 2;
      description "reserved1";
    }
    enum reserved2 {
      value 3;
      description "reserved2";
    }
  }
  mandatory true;
  description "This is a 2-bit unsigned integer value. It defines whether a service data flow can lose the resources assigned to it in order to admit a service data flow with a higher priority level.";
}

typedef Aggregate-Max-DL-Bit-Rate-Value {
  type uint32;
  description "The aggregate maximum downlink bit rate that is requested/allocated for downlink IP flows. The measurement units are bits per second.";
}

typedef Aggregate-Max-UL-Bit-Rate-Value {
  type uint32;
  description "The aggregate maximum downlink bit rate that is requested/allocated for downlink IP flows. The measurement units are bits per second.";
}

typedef Guaranteed-DL-Bit-Rate-Value {
  type uint32;
}
typedef Guaranteed-UL-Bit-Rate-Value {
    type uint32;
    description
        "The guaranteed bandwidth in bits per second for uplink
         IP flows. The measurement units are bits per second.";
}

grouping qosattribute {
    leaf per-mn-agg-max-dl {
        type qos-pmip:Per-MN-Agg-Max-DL-Bit-Rate-Value;
        description "Per-MN-Agg-Max-DL-Bit-Rate Value";
    }
    leaf per-mn-agg-max-ul {
        type qos-pmip:Per-MN-Agg-Max-UL-Bit-Rate-Value;
        description "Per-MN-Agg-Max-UL-Bit-Rate Value";
    }
}
container per-session-agg-max-dl {
    uses qos-pmip:Per-Session-Agg-Max-Bit-Rate-Value;
    description "Per-Session-Agg-Max-Bit-Rate Value";
}

container per-session-agg-max-ul {
    uses qos-pmip:Per-Session-Agg-Max-Bit-Rate-Value;
    description "Per-Session-Agg-Max-Bit-Rate Value";
}

uses qos-pmip:Allocation-Retention-Priority-Value;
leaf agg-max-dl {
    type qos-pmip:Aggregate-Max-DL-Bit-Rate-Value;
    description "Aggregate-Max-DL-Bit-Rate Value";
}

leaf agg-max-ul {
    type qos-pmip:Aggregate-Max-UL-Bit-Rate-Value;
    description "Aggregate-Max-UL-Bit-Rate Value";
}

leaf gbr-dl {
    type qos-pmip:Guaranteed-DL-Bit-Rate-Value;
    description "Guaranteed-DL-Bit-Rate Value";
}

leaf gbr-ul {
    type qos-pmip:Guaranteed-UL-Bit-Rate-Value;
    description "Guaranteed-UL-Bit-Rate Value";
}

description "PMIP QoS Attributes. Note Vendor option is not a part of this grouping";

grouping qosoption {
    leaf srid {
        type sr-id;
        mandatory true;
        description "Service Request Identifier";
    }

    leaf trafficclass {
        type traffic-class;
        mandatory true;
        description "Traffic Class";
    }

    leaf operationcode {
        type operational-code;
        mandatory true;
        description "Operation Code";
    }

    uses qos-pmip:qosattribute;
    uses qos-pmip:QoS-Vendor-Specific-Attribute-Value-Base;
container traffic-selector {
    uses traffic-selectors:traffic-selector;
    description "traffic selector";
    description "PMIP QoS Option";
}

A.4. Traffic Selectors YANG Model

This module defines traffic selector types commonly used in Proxy Mobile IP (PMIP).

This module references [RFC6991].

<CODE BEGINS> file "ietf-trafficselector-types@2018-05-17.yang"
module ietf-trafficselector-types {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-trafficselector-types";
    prefix "traffic-selectors";
    import ietf-inet-types {
        prefix inet;
        revision-date 2013-07-15;
    }
    organization "IETF Distributed Mobility Management (DMM) Working Group";
    contact "WG Web: <http://tools.ietf.org/wg/netmod/>
    WG List: <mailto:netmod@ietf.org>
    WG Chair: Dapeng Liu
    <mailto:maxpassion@gmail.com>
    WG Chair: Sri Gundavelli
    <mailto:sgundave@cisco.com>
    Editor: Satoru Matsushima
    <mailto:satoru.matsushima@softbank.co.jp>
    Editor: Lyle Bertz
    <mailto:lylebe551144@gmail.com>";
description
"This module contains a collection of YANG definitions for traffic selectors for flow bindings.

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

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

revision 2018-05-17 {
    description
        "Initial Revision.";
    reference
        "RFC 6088: Traffic Selectors for Flow Bindings";
}

// Identities
identity traffic-selector-format {
    description
        "The base type for Traffic-Selector Formats";
}

identity ipv4-binary-selector-format {
    base traffic-selector-format;
    description
        "IPv4 Binary Traffic Selector Format";
}

identity ipv6-binary-selector-format {
    base traffic-selector-format;
    description
        "IPv6 Binary Traffic Selector Format";
}

// Type definitions and groupings
typedef ipsec-spi {
    type uint32;
    description
        "The first 32-bit IPsec Security Parameter Index (SPI) value on data. This field is defined in [RFC4303].";
}

grouping traffic-selector-base {
  description "A grouping of the common leaves between the v4 and v6 Traffic Selectors";
  container ipsec-spi-range {
    presence "Enables setting ipsec spi range";
    description "Inclusive range representing IPSec Security Parameter Indices to be used. When only start-spi is present, it represents a single spi.";
    leaf start-spi {
      type ipsec-spi;
      mandatory true;
      description "The first 32-bit IPsec SPI value on data.";
    }
    leaf end-spi {
      type ipsec-spi;
      must ". >= ../start-spi" {
        error-message "The end-spi must be greater than or equal to start-spi";
      }
      description "If more than one contiguous SPI value needs to be matched, then this field indicates the end value of a range.";
    }
  }
  container source-port-range {
    presence "Enables setting source port range";
    description "Inclusive range representing source ports to be used. When only start-port is present, it represents a single port. These value(s) are from the range of port numbers defined by IANA (http://www.iana.org).";
    leaf start-port {
      type inet:port-number;
      mandatory true;
      description "The first 16-bit source port number to be matched";
    }
    leaf end-port {
      type inet:port-number;
      must ". >= ../start-port" {
error-message
"The end-port must be greater than or equal to start-port";
}

description
"The last 16-bit source port number to be matched"
}

container destination-port-range {
presence "Enables setting destination port range";
description
"Inclusive range representing destination ports to be used. When only start-port is present, it represents a single port.";
leaf start-port {
  type inet:port-number;
  mandatory true;
description
  "The first 16-bit destination port number to be matched";
}
leaf end-port {
  type inet:port-number;
must ". >= ../start-port" {
    error-message
    "The end-port must be greater than or equal to start-port";
  }
description
  "The last 16-bit destination port number to be matched";
}
}


grouping ipv4-binary-traffic-selector {
  container source-address-range-v4 {
presence "Enables setting source IPv4 address range";
description
  "Inclusive range representing IPv4 addresses to be used. When only start-address is present, it represents a single address.";
leaf start-address {
  type inet:ipv4-address;
  mandatory true;
description
  "The first source address to be matched";
}
leaf end-address {
  type inet:ipv4-address;
description
  "The last source address to be matched";
}
}
"The last source address to be matched";
}
}

container destination-address-range-v4 {
  presence "Enables setting destination IPv4 address range";
  description
    "Inclusive range representing IPv4 addresses to be used. When only start-address is present, it represents a single address."
  leaf start-address {
    type inet:ipv4-address;
    mandatory true;
    description
      "The first destination address to be matched"
  }
  leaf end-address {
    type inet:ipv4-address;
    description
      "The last destination address to be matched"
  }
}

container ds-range {
  presence "Enables setting dscp range";
  description
    "Inclusive range representing DiffServ Codepoints to be used. When only start-ds is present, it represents a single Codepoint."
  leaf start-ds {
    type inet:dscp;
    mandatory true;
    description
      "The first differential service value to be matched"
  }
  leaf end-ds {
    type inet:dscp;
    must ". >= ../start-ds" {
      error-message
        "The end-ds must be greater than or equal to start-ds"
    }
    description
      "The last differential service value to be matched"
  }
}

container protocol-range {
  presence "Enables setting protocol range";
  description
    "Inclusive range representing IP protocol(s) to be used. When only start-protocol is present, it represents a single
protocol.");
leaf start-protocol {
type uint8;
mandatory true;
description "The first 8-bit protocol value to be matched.";
}
leaf end-protocol {
type uint8;
must ". >= ../start-protocol" {
  error-message
    "The end-protocol must be greater than or equal to start-protocol";
}
description "The last 8-bit protocol value to be matched.";
}
description "ipv4 binary traffic selector";
}


grouping ipv6-binary-traffic-selector {
container source-address-range-v6 {
presence "Enables setting source IPv6 address range";
description "Inclusive range representing IPv6 addresses to be used. When only start-address is present, it represents a single address.";
leaf start-address {
type inet:ipv6-address;
mandatory true;
description "The first source address, from the range of 128-bit IPv6 addresses to be matched";
}
leaf end-address {
type inet:ipv6-address;
description "The last source address, from the range of 128-bit IPv6 addresses to be matched";
}
}
container destination-address-range-v6 {
presence "Enables setting destination IPv6 address range";
description "Inclusive range representing IPv6 addresses to be used. When only start-address is present, it represents a single address.";
leaf start-address {

type inet:ipv6-address;
mandatory true;
description
"The first destination address, from the range of 128-bit IPv6 addresses to be matched";
}
leaf end-address {
  type inet:ipv6-address;
description
"The last destination address, from the range of 128-bit IPv6 addresses to be matched";
}
}
container flow-label-range {
presence "Enables setting Flow Label range";
description
"Inclusive range representing IPv4 addresses to be used. When only start-flow-label is present, it represents a single flow label.";
leaf start-flow-label {
  type inet:ipv6-flow-label;
description
"The first flow label value to be matched";
}
leaf end-flow-label {
  type inet:ipv6-flow-label;
  must ". >= ../start-flow-label" {
    error-message
    "The end-flow-lable must be greater than or equal to start-flow-label";
  }
  description
  "The first flow label value to be matched";
}
}
container traffic-class-range {
presence "Enables setting the traffic class range";
description
"Inclusive range representing IPv4 addresses to be used. When only start-traffic-class is present, it represents a single traffic class.";
leaf start-traffic-class {
  type inet:dscp;
  description
  "The first traffic class value to be matched";
  reference
  "RFC 3260: New Terminology and Clarifications for Diffserv"
  "RFC 3168: The Addition of Explicit Congestion Notification"
(ECN) to IP*;
}

leaf end-traffic-class {
  type inet:dscp;
  must ".. >=../start-traffic-class" {
    error-message
    "The end-traffic-class must be greater than or equal to
    start-traffic-class";
  }
  description
  "The last traffic class value to be matched";
}

container next-header-range {
  presence "Enables setting Next Header range";
  description
  "Inclusive range representing Next Headers to be used. When
  only start-next-header is present, it represents a
  single Next Header."
  leaf start-next-header {
    type uint8;
    description
    "The first 8-bit next header value to be matched.";
  }
  leaf end-next-header {
    type uint8;
    must ".. >=../start-next-header" {
      error-message
      "The end-next-header must be greater than or equal to
      start-next-header";
    }
    description
    "The last 8-bit next header value to be matched.";
  }
}

description "ipv6 binary traffic selector";

grouping traffic-selector {
  leaf ts-format {
    type identityref {
      base traffic-selector-format;
    }
    description "Traffic Selector Format";
  }
  uses traffic-selectors:traffic-selector-base;
  uses traffic-selectors:ipv4-binary-traffic-selector;
  uses traffic-selectors:ipv6-binary-traffic-selector;
description
"The traffic selector includes the parameters used to match
packets for a specific flow binding."
reference
"RFC 6089: Flow Bindings in Mobile IPv6 and Network
Mobility (NEMO) Basic Support";
}

A.5. RFC 5777 Classifier YANG Model

This module defines the RFC 5777 Classifier.

This module references [RFC5777].

<CODE BEGINS> file "ietf-diam-trafficclassifier@2018-05-17.yang"
module ietf-diam-trafficclassifier {
  yang-version 1.1;

  namespace
"urn:ietf:params:xml:ns:yang:ietf-diam-trafficclassifier";

  prefix "diamclassifier";

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

  import ietf-yang-types { prefix yang-types; }

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

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

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

WG Chair: Sri Gundavelli
<mailto:sgundave@cisco.com>

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

Editor: Lyle Bertz

typedef eui64-address-type {
  type string {
    length "6";
  }
  description
  "specifies a single layer 2 address in EUI-64 format. The value is an 8-octet encoding of the address as it would appear in the frame header."
}

typedef direction-type {
  type enumeration {
    enum IN {
      value 0;
      description
      "Applies to flows from the managed terminal."
    }
    enum OUT {
      value 1;
      description
      "Applies to flows to the managed terminal."
    }
  }
}


definitions

    enum BOTH {
        value 2;
        description "Applies to flows both to and from the managed
terminal.";
    }

description

    "Specifies in which direction to apply the classifier.";

typedef negated-flag-type {
    type enumeration {
        enum False { value 0;
            description "false"; }
        enum True { value 1;
            description "True"; }
    }

description

    "When set to True, the meaning of the match is
inverted and the classifier will match addresses
other than those specified by the From-Spec or
To-Spec AVP.

    Note that the negation does not impact the port
comparisons.";

grouping index {
    leaf index {
        type uint16;
        mandatory true;
        description "Identifier used for referencing";
    }

description "Index Value";

grouping to-from-spec-value {
    leaf-list ip-address {
        type inet:ip-address;
        description "IP address";
    }
}

list ip-address-range {
    key index;
    uses diamclassifier:index;
    leaf ip-address-start {
        type inet:ip-address;
        description "IP Address Start";
    }
    leaf ip-address-end {
        type inet:ip-address;
description "IP Address End";
}
description "IP Address Range";
leaf-list ip-address-mask {
  type inet:ip-prefix;
  description "IP Address Mask";
}
leaf-list mac-address {
  type yang-types:mac-address;
  description "MAC address";
}
list mac-address-mask {
  key mac-address;
  leaf mac-address {
    type yang-types:mac-address;
    mandatory true;
    description "MAC address";
  }
  leaf macaddress-mask-pattern {
    type yang-types:mac-address;
    mandatory true;
    description "The value specifies the bit positions of a MAC address that are taken for matching.";
  }
  description "MAC Address Mask";
}
leaf-list eui64-address {
  type diamclassifier:eui64-address-type;
  description "EUI64 Address";
}
list eui64-address-mask {
  key eui64-address;
  leaf eui64-address {
    type diamclassifier:eui64-address-type;
    mandatory true;
    description "eui64 address";
  }
  leaf eui64-address-mask-pattern {
    type diamclassifier:eui64-address-type;
    mandatory true;
    description "The value is 8 octets specifying the bit positions of a EUI64 address that are taken for matching.";
  }
  description "EUI64 Address Mask";
}
leaf-list port {
    type inet:port-number;
    description "Port Number";
}

list port-range {
    key index;
    uses diamclassifier:index;
    leaf ip-address-start {
        type inet:port-number;
        description "Port Start";
    }
    leaf ip-address-end {
        type inet:port-number;
        description "Port End";
    }
    description "Port Range";
}

leaf negated {
    type diamclassifier:negated-flag-type;
    description "Negated";
}

leaf use-assigned-address {
    type boolean;
    description "Use Assigned Address";
}

description "Basic traffic description value";


grouping option-type-group {
    leaf option-type {
        type uint8;
        mandatory true;
        description "Option Type";
    }
    leaf-list ip-option-value {
        type string;
        description "Option Value";
    }
    leaf negated {
        type diamclassifier:negated-flag-type;
        description "Negated";
    }

description "Common X Option Pattern";
}
typedef vlan-id {
    type uint32 {
range "0..4095";
}
description "VLAN ID";
}
grouping classifier {
leaf protocol {
    type uint8;
    description "Protocol";
}
leaf direction {
    type diamclassifier:direction-type;
    description "Direction";
}
list from-spec {
    key index;
    uses diamclassifier:index;
    uses diamclassifier:to-from-spec-value;
    description "from specification";
}
list to-spec {
    key index;
    uses diamclassifier:index;
    uses diamclassifier:to-from-spec-value;
    description "to specification";
}
leaf-list disffserv-code-point {
    type inet:dscp;
    description "DSCP";
}
leaf fragmentation-flag {
    type enumeration {
        enum DF {
            value 0;
            description "Don’t Fragment";
        }
        enum MF {
            value 1;
            description "More Fragments";
        }
    }
    description "Fragmenttation Flag";
}
list ip-option {
    key option-type;
    uses diamclassifier:option-type-group;
    description "IP Option Value";
}
list tcp-option {
  key option-type;
  uses diamclassifier:option-type-group;
  description "TCP Option Value";
}
list tcp-flag {
  key tcp-flag-type;
  leaf tcp-flag-type {
    type uint32;
    mandatory true;
    description "TCP Flag Type";
  }
  leaf negated {
    type diamclassifier:negated-flag-type;
    description "Negated";
  }
  description "TCP Flags";
}
list icmp-option {
  key option-type;
  uses diamclassifier:option-type-group;
  description "ICMP Option Value";
}
list eth-option {
  key index;
  uses diamclassifier:index;
  container eth-proto-type {
    leaf-list eth-ether-type {
      type string {
        length "2";
      }
      description "value of ethertype field";
    }
    leaf-list eth-sap {
      type string {
        length "2";
      }
      description "802.2 SAP";
    }
    description "Ether Proto Type";
  }
list vlan-id-range {
  key index;
  uses diamclassifier:index;
  leaf-list s-vlan-id-start {
    type diamclassifier:vlan-id;
    description "S-VID VLAN ID Start";
  }
}
leaf-list s-vlan-id-end {
type diamclassifier:vlan-id;
description "S-VID VLAN ID End";
}
leaf-list c-vlan-id-start {
type diamclassifier:vlan-id;
description "C-VID VLAN ID Start";
}
leaf-list c-vlan-id-end {
type diamclassifier:vlan-id;
description "C-VID VLAN ID End";

description "VLAN ID Range";
}
list user-priority-range {
key index;
uses diamclassifier:index;
leaf-list low-user-priority {
type uint32 {
  range "0..7";
}
description "Low User Priority";
}
leaf-list high-user-priority {
type uint32 {
  range "0..7";
}
description "High User Priority";
}
description "User priority range";
}
description "Ether Option";
}
description "RFC 5777 Classifier";
}

Appendix B. FPC YANG Tree Structure

This section only shows the structure for FPC YANG model. NOTE, it does NOT show the settings, Action values or Descriptor Value.

descriptor_value:
  +++-rw (descriptor-value)
  +--+:(all-traffic)
  |   ++-rw all-traffic?
  |     empty
  +--+:(no-traffic)
no-traffic? empty

(prefix-descriptor)

destination-ip? inet:ip-prefix
source-ip? inet:ip-prefix

(pmip-selector)

ts-format? identityref

ipsec-spi-range!

start-spi ipsec-spi
end-spi? ipsec-spi

source-port-range!

start-port inet:port-number
end-port? inet:port-number

destination-port-range!

start-port inet:port-number
end-port? inet:port-number

source-address-range-v4!

start-address inet:ipv4-address
end-address? inet:ipv4-address

destination-address-range-v4!

start-address inet:ipv4-address
end-address? inet:ipv4-address

ds-range!

start-ds inet:dscp
end-ds? inet:dscp

protocol-range!

start-protocol uint8
end-protocol? uint8

source-address-range-v6!

start-address inet:ipv6-address
end-address? inet:ipv6-address

destination-address-range-v6!

start-address inet:ipv6-address
end-address? inet:ipv6-address

flow-label-range!

start-flow-label? inet:ipv6-flow-label
end-flow-label? inet:ipv6-flow-label

traffic-class-range!

start-traffic-class? inet:dscp
end-traffic-class? inet:dscp

next-header-range!

start-next-header? uint8
end-next-header? uint8

(rfc5777-classifier-template)

protocol? uint8
direction? diamclassifier:direction-type

from-spec* [index]

index uint16
---rw ip-address* inet:ip-address
  ---rw ip-address-range* [index]
    ---rw index uint16
    ---rw ip-address-start? inet:ip-address
    ---rw ip-address-end? inet:ip-address
  ---rw ip-address-mask* inet:ip-prefix
  ---rw mac-address* yang-types:mac-address
  ---rw mac-address-mask* [mac-address]
    ---rw mac-address yang-types:mac-address
    ---rw mac-address-mask-pattern yang-types:mac-address
  ---rw eui64-address*
    diamclassifier:eui64-address-type
  ---rw eui64-address-mask* [eui64-address]
    ---rw eui64-address diamclassifier:eui64-address-type
  ---rw eui64-address-mask-pattern
diamclassifier:eui64-address-type
  ---rw port* inet:port-number
  ---rw port-range* [index]
    ---rw index uint16
    ---rw ip-address-start? inet:port-number
    ---rw ip-address-end? inet:port-number
  ---rw negated?
diamclassifier:negated-flag-type
  ---rw to-spec* [index]
    ---rw index uint16
    ---rw ip-address* inet:ip-address
    ---rw ip-address-range* [index]
      ---rw index uint16
      ---rw ip-address-start? inet:ip-address
      ---rw ip-address-end? inet:ip-address
    ---rw ip-address-mask* inet:ip-prefix
    ---rw mac-address* yang-types:mac-address
    ---rw mac-address-mask* [mac-address]
      ---rw mac-address yang-types:mac-address
      ---rw mac-address-mask-pattern yang-types:mac-address
    ---rw eui64-address*
      diamclassifier:eui64-address-type
    ---rw eui64-address-mask* [eui64-address]
      ---rw eui64-address diamclassifier:eui64-address-type
    ---rw eui64-address-mask-pattern
diamclassifier:eui64-address-type
    ---rw port* inet:port-number
    ---rw port-range* [index]
      ---rw index uint16
      ---rw ip-address-start? inet:port-number
++:(ipv4-ipv6-remote)
  | ++-rw ipv4-ipv6-remote? inet:ip-address
++:(ipv6-prefix-remote)
  | ++-rw ipv6-prefix-remote? inet:ipv6-prefix
++:(next-header)
  | ++-rw next-header? uint8
++:(local-port)
  | ++-rw local-port? inet:port-number
++:(local-port-range)
  | ++-rw local-port-lo? inet:port-number
  | ++-rw local-port-hi? inet:port-number
++:(remote-port)
  | ++-rw remote-port? inet:port-number
++:(remote-port-range)
  | ++-rw remote-port-lo? inet:port-number
  | ++-rw remote-port-hi? inet:port-number
++:(ipsec-index)
  | ++-rw ipsec-index? traffic-selectors:ipsec-spi
++:(traffic-class)
  | ++-rw traffic-class? inet:dscp
++:(traffic-class-range)
  | ++-rw traffic-class-lo? inet:dscp
  | ++-rw traffic-class-hi? inet:dscp
++:(flow-label)
  | ++-rw flow-label* inet:ipv6-flow-label
++:(tunnel-info)
  ++-rw tunnel-info
    | ++-rw tunnel-local-address? inet:ip-address
    | ++-rw tunnel-remote-address? inet:ip-address
    | ++-rw mtu-size? uint32
    | ++-rw tunnel? identityref
    | ++-rw payload-type? enumeration
    | ++-rw gre-key? uint32
++-gw gtp-tunnel-info
    | ++-rw local-tunnel-identifier? uint32
    | ++-rw remote-tunnel-identifier? uint32
    | ++-rw sequence-numbers-enabled? boolean
++-rw ebi? fpcbase:ebi-type
++-rw lbi? fpcbase:ebi-type

action_value:
++:(action-value)
  ++-rw (action-value)
    | ++-:(drop)
      | | ++-rw drop? empty
    | ++-:(rewrite)
      | | ++-rw rewrite
        | | ++-rw (rewrite-value)?
route descriptor
    +--rw destination-ip? inet:ip-prefix
    +--rw source-ip? inet:ip-prefix
route ipsec-spi-range!
    +--rw start-spi ipsec-spi
    +--rw end-spi? ipsec-spi
route source-port-range!
    +--rw start-port inet:port-number
    +--rw end-port? inet:port-number
route destination-port-range!
    +--rw start-port inet:port-number
    +--rw end-port? inet:port-number
route source-address-range-v4!
    +--rw start-address inet:ipv4-address
    +--rw end-address? inet:ipv4-address
route destination-address-range-v4!
    +--rw start-address inet:ipv4-address
    +--rw end-address? inet:ipv4-address
route ds-range!
    +--rw start-ds inet:dscp
    +--rw end-ds? inet:dscp
route protocol-range!
    +--rw start-protocol uint8
    +--rw end-protocol? uint8
route source-address-range-v6!
    +--rw start-address inet:ipv6-address
    +--rw end-address? inet:ipv6-address
route destination-address-range-v6!
    +--rw start-address inet:ipv6-address
    +--rw end-address? inet:ipv6-address
route flow-label-range!
    +--rw start-flow-label? inet:ipv6-flow-label
    +--rw end-flow-label? inet:ipv6-flow-label
route traffic-class-range!
    +--rw start-traffic-class? inet:dscp
    +--rw end-traffic-class? inet:dscp
route next-header-range!
    +--rw start-next-header? uint8
    +--rw end-next-header? uint8
route classifier-template
    +--rw protocol? uint8
    +--rw direction? diamclassifier:direction-type
    +--rw from-spec* [index]
        +--rw index uint16
diamclassifier:eui64-address-type
  +--rw port*                    inet:port-number
  +--rw port-range* [index]
    +--rw index                   uint16
    +--rw ip-address-start?       inet:port-number
    +--rw ip-address-end?        inet:port-number
  +--rw negated?
    diamclassifier:negated-flag-type
    +--rw use-assigned-address?  boolean
  +--rw diffserv-code-point*     inet:dscp
  +--rw fragmentation-flag?     enumeration
  +--rw ip-option* [option-type]
    +--rw option-type            uint8
    +--rw ip-option-value*       string
    +--rw negated?
      diamclassifier:negated-flag-type
  +--rw tcp-option* [option-type]
    +--rw option-type            uint8
    +--rw ip-option-value*       string
    +--rw negated?
      diamclassifier:negated-flag-type
  +--rw tcp-flag* [tcp-flag-type]
    +--rw tcp-flag-type          uint32
    +--rw negated?
      diamclassifier:negated-flag-type
  +--rw icmp-option* [option-type]
    +--rw option-type            uint8
    +--rw ip-option-value*       string
    +--rw negated?
      diamclassifier:negated-flag-type
  +--rw eth-option* [index]
    +--rw index                  uint16
    +--rw eth-prototyoe
      +--rw eth-ether-type*      string
      +--rw eth-sap*            string
    +--rw vlan-id-range* [index]
      +--rw index                 uint16
      +--rw s-vlan-id-start*      diamclassifier:vlan-id
      +--rw s-vlan-id-end*       diamclassifier:vlan-id
      +--rw c-vlan-id-start*      diamclassifier:vlan-id
      +--rw c-vlan-id-end*       diamclassifier:vlan-id
    +--rw user-priority-range* [index]
      +--rw index                  uint16
      +--rw low-user-priority*    uint32
+--rw mpls-path?                 fpcbase:fpc-mpls-label
  +--:(nsh)
    +--rw nsh?                     string
  +--:(interface)
    +--rw interface?               uint16
  +--:(segment-identifier)
    +--rw segment-identifier?      fpcbase:segment-id
  +--:(mpls-label-stack)
    +--rw mpls-label-stack*       fpcbase:fpc-mpls-label
  +--:(mpls-sr-stack)
    +--rw mpls-sr-stack*           fpcbase:fpc-mpls-label
  +--:(srv6-stack)
    +--rw srv6-stack*              fpcbase:segment-id
  +--:(tunnel-info)
    +--rw tunnel-info
      +--rw tunnel-local-address?   inet:ip-address
      +--rw tunnel-remote-address?  inet:ip-address
      +--rw mtu-size?                uint32
      +--rw tunnel?                  identityref
      +--rw payload-type?            enumeration
      +--rw gre-key?                 uint32
      +--rw gtp-tunnel-info
        +--rw local-tunnel-identifier?  uint32
        +--rw remote-tunnel-identifier? uint32
        +--rw sequence-numbers-enabled? boolean
      +--rw ebi?                      fpcbase:ebi-type
      +--rw lbi?                      fpcbase:ebi-type
  +--:(qos)
    +--rw trafficclass?            inet:dscp
    +--rw per-mn-agg-max-dl?
      +--rw qos-pmip:Per-MN-Agg-Max-DL-Bit-Rate-Value
    +--rw per-mn-agg-max-ul?
      +--rw qos-pmip:Per-MN-Agg-Max-UL-Bit-Rate-Value
    +--rw per-session-agg-max-dl
      +--rw max-rate                uint32
      +--rw service-flag            boolean
      +--rw exclude-flag            boolean
    +--rw per-session-agg-max-ul
      +--rw max-rate                uint32
      +--rw service-flag            boolean
      +--rw exclude-flag            boolean
    +--rw priority-level           uint8
    +--rw preemption-capability    enumeration
    +--rw preemption-vulnerability  enumeration
    +--rw agg-max-dl?
      +--rw qos-pmip:Aggregate-Max-DL-Bit-Rate-Value
    +--rw agg-max-ul?
      +--rw qos-pmip:Aggregate-Max-UL-Bit-Rate-Value
++rw gbr-dl?
    qos-pmip:Guaranteed-DL-Bit-Rate-Value
++rw gbr-ul?
    qos-pmip:Guaranteed-UL-Bit-Rate-Value
++rw qci?
    fpcbase:fpc-qos-class-identifier
++rw ue-agg-max-bitrate?  uint32
++rw apn-ambr?  uint32

policy-configuration-value:
++rw (policy-configuration-value)?
    +++:(descriptor-value)
    | ...
    +++:(action-value)
    | ...
    +++:(setting-value)
    | ++rw setting?  <anydata>

policy-configuration:
++rw policy-configuration* [index]
    ++rw index  uint16
    ++rw extensible?  boolean
    ++rw static-attributes*  string
    ++rw mandatory-attributes*  string
    ++rw entity-state?  enumeration
    ++rw version?  uint32
    ++rw (policy-configuration-value)?
    | ...

module: ietf-dmm-fpc
++rw tenant* [tenant-key]
    ++rw tenant-key  fpc:fpc-identity
++rw topology-information-model
++rw service-group* [service-group-key role-key]
    ++rw service-group-key  fpc:fpc-identity
    ++rw service-group-name?  string
    ++rw role-key  identityref
    ++rw role-name?  string
    ++rw protocol*  identityref
    ++rw feature*  identityref
    ++rw service-group-configuration* [index]
        ++rw index  uint16
        | ++rw (policy-configuration-value)?
        | | ...
    ++rw dpn* [dpn-key]
        ++rw dpn-key  fpc:fpc-identity
        ++rw referenced-interface* [interface-key]
            | ++rw interface-key  fpc:fpc-identity
---rw peer-service-group-key* fpc:fpc-identity

---rw dpn* [dpn-key]
  +--rw dpn-key fpc:fpc-identity
  +--rw dpn-name? string
  +--rw dpn-resource-mapping-reference? string
  +--rw domain-key fpc:fpc-identity
  +--rw service-group-key* fpc:fpc-identity
  +--rw interface* [interface-key]
    +--rw interface-key fpc:fpc-identity
    +--rw interface-name? string
    +--rw role? identityref
    +--rw protocol* identityref
    +--rw interface-configuration* [index]
      +--rw (policy-configuration-value)?
      |  ...
  +--rw dpn-policy-configuration* [policy-template-key]
    +--rw policy-template-key fpc:fpc-identity
    +--rw policy-configuration* [index]
      +--rw index uint16
      +--rw (policy-configuration-value)?
      |  ...
  +--rw domain* [domain-key]
    +--rw domain-key fpc:fpc-identity
    +--rw domain-name? string
    +--rw domain-policy-configuration* [policy-template-key]
      +--rw policy-template-key fpc:fpc-identity
      +--rw policy-configuration* [index]
      |  ...
  +--rw dpn-checkpoint
    +--rw basename? fpc:fpc-identity
    +--rw base-checkpoint? string
  +--rw service-group-checkpoint
    +--rw basename? fpc:fpc-identity
    +--rw base-checkpoint? string
  +--rw dpn-checkpoint
    +--rw basename? fpc:fpc-identity
    +--rw base-checkpoint? string
  +--rw policy-information-model
    +--rw action-template* [action-template-key]
      +--rw action-template-key fpc:fpc-identity
      +--rw (action-value)
      |  ...
      +--rw extensible? boolean
      +--rw static-attributes* string
      +--rw mandatory-attributes* string
      +--rw entity-state? enumeration
      +--rw version? uint32
      +--rw descriptor-template* [descriptor-template-key]
|  +--RW descriptor-template-key             fpc:fpc-identity |
|  +--RW (descriptor-value)                 ... |
|  +--RW extensible?                       boolean |
|  +--RW static-attributes*                string |
|  +--RW mandatory-attributes*             string |
|  +--RW entity-state?                     enumeration |
|  +--RW version?                          uint32 |
|  +--RW rule-template*                    [rule-template-key] |
|   +--RW descriptor-match-type            enumeration |
|   +--RW descriptor-configuration*        [descriptor-template-key] |
|     +--RW descriptor-template-key        fpc:fpc-identity |
|     +--RW direction?                     rfc5777:direction-type |
|     +--RW setting?                       <anydata> |
|     +--RW attribute-expression*          [index] |
|       +--RW index                        uint16 |
|       +--RW (descriptor-value)           ... |
|   +--RW rule-template*                   [rule-template-key] |
|    +--RW rule-template-key               fpc:fpc-identity |
|    +--RW action-template-key             fpc:fpc-identity |
|    +--RW setting?                        <anydata> |
|    +--RW attribute-expression*           [index] |
|       +--RW index                        uint16 |
|       +--RW (action-value)               ... |
|   +--RW extensible?                      boolean |
|   +--RW static-attributes*               string |
|   +--RW mandatory-attributes*            string |
|   +--RW entity-state?                    enumeration |
|   +--RW version?                         uint32 |
|   +--RW rule-configuration*              [index] |
|      +--RW index                         uint16 |
|      +--RW (policy-configuration-value)? ... |
|   +--RW policy-template*                 [policy-template-key] |
|    +--RW policy-template-key             fpc:fpc-identity |
|    +--RW rule-template*                  [precedence] |
|     +--RW precedence                     uint32 |
|     +--RW rule-template-key              fpc:fpc-identity |
|     +--RW extensible?                    boolean |
|     +--RW static-attributes*             string |
|     +--RW mandatory-attributes*          string |
|     +--RW entity-state?                  enumeration |
|     +--RW version?                       uint32 |
|     +--RW policy-configuration*          [index] |
|      ...
rpcs:
  +--x configure
  +---w input
    +---w client-id          fpc:client-identifier
    +---w execution-delay?   uint32
    +---w yang-patch
      +---w patch-id        string
      +---w comment?        string
      +---w edit*           [edit-id]
        +---w edit-id       string
        +---w operation      enumeration
        +---w target         target-resource-offset
        +---w point?         target-resource-offset
        +---w where?         enumeration
        +---w value?         <anydata>
        +---w reference-scope? fpc:ref-scope
        +---w command-set
          +---w (instr-type)?
            +---w instr-3gpp-mob? fpcbase:threegpp-instr
            +---w instr-pmip?    pmip-commandset
  +---ro output
    +---ro yang-patch-status
    +---ro patch-id        string
    +---ro (global-status)?
      +---(global-errors)
        +---ro errors
          +---ro error*
            +---ro error-type     enumeration
            +---ro error-tag      string
            +---ro error-app-tag? string
            +---ro error-path?    instance-identifier
            +---ro error-message? string
            +---ro error-info?    <anydata>
          +---w ok?            empty

```
++-ro edit-status
  ++-ro edit* [edit-id]
    ++-ro edit-id string
    ++-ro (edit-status-choice)?
      +=:(ok)
        ++-ro ok? empty
        +=-ro notify-follows? boolean
      +=-ro (edit-status-choice)?
        +=-ro ok? empty
        ++-ro notify-follows? boolean
        +=-ro subsequent-edit* [edit-id]
          ++-ro edit-id string
          ++-ro operation enumeration
          +=-ro target
            ypatch:target-resource-offset
          +=-ro point?
            ypatch:target-resource-offset
          +=-ro where? enumeration
          +=-ro value? <anydata>
      +=-ro (errors)
        ++-ro errors
          +=-ro error* 
            +=-ro error-type enumeration
            +=-ro error-tag string
            +=-ro error-app-tag? string
            +=-ro error-path?
              instance-identifier
            +=-ro error-message? string
            +=-ro error-info? <anydata>
        +=x register_monitor
          ++w input
            +=w client-id fpc:client-identifier
            +=w execution-delay? uint32
            +=w operation-id uint64
            +=w monitor* [monitor-key]
              +=w extensible? boolean
              +=w static-attributes* string
              +=w mandatory-attributes* string
              +=w entity-state? enumeration
              +=w version? uint32
              +=w monitor-key fpc:fpc-identity
              +=w target? string
              +=w deferrable? boolean
            +=w (configuration)
              +=-:(period)
                +=w period? uint32
              +=-:(threshold-config)
                +=w low? uint32
                +=w hi? uint32
              +=-:(schedule)
                +=w schedule? uint32
```

<table>
<thead>
<tr>
<th>ro operation-id</th>
<th>uint64</th>
</tr>
</thead>
<tbody>
<tr>
<td>(edit-status-choice)?</td>
<td></td>
</tr>
<tr>
<td>ro ok?</td>
<td>empty</td>
</tr>
<tr>
<td>:ro errors</td>
<td></td>
</tr>
<tr>
<td>ro error*</td>
<td></td>
</tr>
<tr>
<td>:ro error-type</td>
<td>enumeration</td>
</tr>
<tr>
<td>:ro error-tag</td>
<td>string</td>
</tr>
<tr>
<td>:ro error-app-tag?</td>
<td>string</td>
</tr>
<tr>
<td>:ro error-path?</td>
<td>instance-identifier</td>
</tr>
<tr>
<td>:ro error-message?</td>
<td>string</td>
</tr>
<tr>
<td>:ro error-info?</td>
<td>&lt;anydata&gt;</td>
</tr>
<tr>
<td>notifications:</td>
<td></td>
</tr>
<tr>
<td>n config-result-notification</td>
<td></td>
</tr>
<tr>
<td>:ro yang-patch-status</td>
<td></td>
</tr>
<tr>
<td>:ro patch-id</td>
<td>string</td>
</tr>
<tr>
<td>:ro (global-status)?</td>
<td></td>
</tr>
<tr>
<td>:ro errors</td>
<td></td>
</tr>
<tr>
<td>:ro error*</td>
<td></td>
</tr>
<tr>
<td>:ro error-type</td>
<td>enumeration</td>
</tr>
<tr>
<td>:ro error-tag</td>
<td>string</td>
</tr>
<tr>
<td>:ro error-app-tag?</td>
<td>string</td>
</tr>
<tr>
<td>:ro error-path?</td>
<td>instance-identifier</td>
</tr>
<tr>
<td>:ro error-message?</td>
<td>string</td>
</tr>
<tr>
<td>:ro error-info?</td>
<td>&lt;anydata&gt;</td>
</tr>
<tr>
<td>:ro ok?</td>
<td>empty</td>
</tr>
<tr>
<td>:ro edit-status</td>
<td></td>
</tr>
<tr>
<td>:ro edit* [edit-id]</td>
<td></td>
</tr>
<tr>
<td>:ro edit-id</td>
<td>string</td>
</tr>
<tr>
<td>:ro (edit-status-choice)?</td>
<td></td>
</tr>
<tr>
<td>:ro errors</td>
<td></td>
</tr>
<tr>
<td>:ro error*</td>
<td></td>
</tr>
<tr>
<td>:ro error-type</td>
<td>enumeration</td>
</tr>
<tr>
<td>:ro error-tag</td>
<td>string</td>
</tr>
<tr>
<td>:ro error-app-tag?</td>
<td>string</td>
</tr>
<tr>
<td>:ro error-path?</td>
<td>instance-identifier</td>
</tr>
<tr>
<td>:ro error-message?</td>
<td>string</td>
</tr>
<tr>
<td>:ro error-info?</td>
<td>&lt;anydata&gt;</td>
</tr>
<tr>
<td>:ro ok?</td>
<td>empty</td>
</tr>
<tr>
<td>:ro subsequent-edit* [edit-id]</td>
<td></td>
</tr>
<tr>
<td>:ro edit-id</td>
<td>string</td>
</tr>
</tbody>
</table>
Appendix C. Change Log

C.1. Changes since Version 09

The following changes have been made since version 09

Migration to a Template based framework. This affects all elements. The framework has a template definition language.

Basenames are split into two aspects. The first is version which applies to Templates. The second is checkpointing which applies to specific sections only.

Rule was inside Policy and now is Rule-Template and stands as a peer structure to Policy.

Types, e.g. Descriptor Types, Action Types, etc., are now templates that have no values filled in.

The embedded rule has been replaced by a template that has no predefined variables. All rules, pre-configured or embedded, are realized as Policy instantiations.

The Unassigned DPN is used to track requests vs. those that are installed, i.e. Agent assignment of Policy is supported.
The Topology system supports selection information by ServiceGroup or ServiceEndpoint.

DPN Peer Groups and DPN Groups are now PeerServiceGroup and ServiceGroup.

Bulk Configuration and Configuration now follow a style similar to YANG Patch. Agents MAY response back with edits it made to complete the Client edit request.

RFC 5777 Classifiers have been added.

All operations have a common error format.

C.2. Changes since Version 10

The following changes have been made since version 10

Service-Endpoints eliminated. Service-Group and DPN interfaces changed to hold information previously held by Service-Endpoint as noted in ML during IETF 101.

Service-Group resides under the Topology-Information-Mode

The Domain now has a checkpoint and the Topology Information Model checkpoint was removed to avoid any overlaps in checkpoints.

Scrubbed YANG for NMNA compliance and Guidelines (RFC 6087bis).

Monitor lifecycle, policy and policy installation examples added.

Authors' Addresses

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

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

Lyle Bertz
6220 Sprint Parkway
Overland Park KS, 66251
USA

Email: lylebe551144@gmail.com
Marco Liebsch
NEC Laboratories Europe
NEC Europe Ltd.
Kurfuersten-Anlage 36
D-69115 Heidelberg
Germany
Phone: +49 6221 4342146
Email: liebsch@neclab.eu

Sri Gundavelli
Cisco
170 West Tasman Drive
San Jose, CA  95134
USA
Email: sgundave@cisco.com

Danny Moses
Email: danny.moses@intel.com

Charles E. Perkins
Futurewei Inc.
2330 Central Expressway
Santa Clara, CA  95050
USA
Phone: +1-408-330-4586
Email: charliep@computer.org
Applications differ with respect to whether they need 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, as described in [RFC7333]. This document defines a new concept of enabling applications to influence the network’s mobility services (session continuity and/or IP address reachability) on a per-Socket basis, and suggests extensions to the networking stack’s API to accommodate this concept.

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 https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on August 26, 2019.

Copyright Notice

Copyright (c) 2019 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 (https://trustee.ietf.org/license-info) in effect on the date of publication of this document.
Introduction

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

- Session Continuity

The ability to maintain an ongoing transport interaction by keeping the same local end-point IP address throughout the life-time of the IP socket despite the mobile host changing its point of attachment.

within the IP network topology. The IP address of the host may change after closing the IP socket and before opening a new one, but that does not jeopardize the ability of applications using these IP sockets to work flawlessly. 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 sessions, and even in the absence of any session. The IP address may be published in a long-term registry (e.g., DNS), and is made available for serving incoming (e.g., TCP) connections. IP address reachability is essential for mobile hosts to use specific/published IP addresses.

Mobile IP is designed to provide both 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 session continuity and IP address reachability.

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

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

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

This document proposes a solution for applications running on mobile hosts to indicate when establishing the network connection (‘on demand’) whether they need session continuity or IP address reachability. The network protocol stack on the mobile host, in conjunction with the network infrastructure, provides the required type of service. It is for the benefit of both the users and the network operators not to engage an extra level of service unless it is absolutely necessary. It is expected that applications and networks compliant with this specification will 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", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119] [RFC8174] when, they appear in all capitals, as shown here.

3. Solution

3.1. High-level Description

Enabling applications to indicate their mobility service requirements e.g. session continuity and/or IP address reachability, comprises the following steps:

- The application indicates to the network stack (local to the mobile host) the desired mobility service.

- The network stack assigns a source IP address based on an IP prefix with the desired services that was previously provided by the network. If such an IP prefix is not available, the network stack performs the additional steps below.

- The network stack sends a request to the network for a new source IP prefix that is associated with the desired mobility service.

- The network responds with the suitable allocated source IP prefix (or responds with a failure indication).
- If the suitable source IP prefix was allocates, the network stack constructs a source IP address and provides it to the application.

This document specifies the new address types associated with mobility services and details the interaction between the applications and the network stack steps. It uses the Socket interface as an example for an API between applications and the network stack. Other steps are outside the scope of this document.

3.2. Types of IP Addresses

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

- Fixed IP Address

A Fixed IP address is an address with a guarantee to be valid for a very long time, regardless of whether it is being used in any packet to/from the mobile host, or whether or not the mobile host is connected to the network, or whether it moves from one point-of-attachment to another (with a different IP prefix) while it is connected.

Fixed IP addresses are required by applications that need both 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 life-time of the socket(s) for which it was requested. It is guaranteed to be valid even after the mobile host had moved from one point-of-attachment to another (with a different IP prefix).

Session-lasting IP addresses are required by applications that need session continuity but do not need IP address reachability.

- Non-persistent IP Address

This type of IP address has no guarantee to exist after a mobile host moves from one point-of-attachment to another, and therefore, no session continuity nor IP address reachability are provided. The IP address is created from an IP prefix that is obtained from the serving IP gateway and is not maintained across gateway changes. In other words, the IP prefix may be released and replaced by a new one when the IP gateway changes due to the movement of the mobile host forcing the creation of a new source IP address with the updated allocated IP prefix.
Graceful Replacement IP Address

In some cases, the network cannot guarantee the validity of the provided IP prefix throughout the duration of the opened socket, but can provide a limited graceful period of time in which both the original IP prefix and a new one are valid. This enables the application some flexibility in the transition from the existing source IP address to the new one.

This gracefulness is still better than the non-persistence type of address for applications that can handle a change in their source IP address but require that extra flexibility.

Applications running as servers at a published IP address require a Fixed IP Address. Long-standing applications (e.g., an SSH session) may also require this type of address. Enterprise applications that connect to an enterprise network via virtual LAN require a Fixed IP Address.

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

Applications with very short sessions, such as DNS clients and instant messengers, can utilize Non-persistent IP Addresses. Even though they could very well use Fixed or Session-lasting IP Addresses, the transmission latency would be minimized when a Non-persistent IP Addresses are used.

Applications that can tolerate a short interruption in connectivity can use the Graceful-replacement IP addresses. For example, a streaming client that has buffering capabilities.

3.3. Granularity of Selection

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

3.4. On Demand Nature

At any point in time, a mobile host may have a combination of IP addresses configured. Zero or more Fixed, zero or more Session-lasting, zero or more Non-persistent and zero or more Graceful-Replacement IP addresses may be configured by the IP stack of the host. The combination may be as a result of the host policy, application demand, or a mix of the two.
When an application requires a specific type of IP address and such an address is not already configured on the host, the IP stack SHALL attempt to configure one. For example, a host may not always have a Session-lasting IP address available. When an application requests one, the IP stack SHALL make an attempt to configure one by issuing a request to the network (see Section 3.5 below for more details). If the operation fails, the IP stack SHALL fail the associated socket request and return an error. If successful, a Session-lasting IP Address gets configured on the mobile host. If another socket requests a Session-lasting IP address at a later time, the same IP address may be served to that socket as well. When the last socket using the same configured IP address is closed, the IP address may be released or kept for future applications that may be launched and require a Session-lasting IP address.

In some cases it might be preferable for the mobile host to request a new Session-lasting IP address for a new opening of an IP socket (even though one was already assigned to the mobile host by the network and might be in use in a different, already active IP sockets). It is outside the scope of this specification to define criteria for choosing to use available addresses or choosing to request new ones. It supports both alternatives (and any combination).

It is outside the scope of this specification to define how the host requests a specific type of prefix and how the network indicates the type of prefix in its advertisement or in its reply to a request.

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

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

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

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

3.5. Conveying the Desired Address Type

[RFC5014] introduced the ability of applications to influence the source address selection with the IPV6_ADDR_PREFERENCE option at the IPPROTO_IPV6 level. This option is used with setsockopt() and getsockopt() calls to set/get address selection preferences.
Extending this further by adding more flags does not work when a request for an address of a certain type results in requiring the IP stack to wait for the network to provide the desired source IP prefix and hence causing the setsockopt() call to block until the prefix is allocated (or an error indication from the network is received).

Alternatively a new socket API is defined - setsc() which allows applications to express their desired type of session continuity service. The new setsc() API will return an IPv6 address that is associated with the desired session continuity service and with status information indicating whether or not the desired service was provided.

An application that wishes to secure a desired service will call setsc() with the service type definition and a place to contain the provided IP address, and call bind() to associate that IP address with the socket (See pseudo-code example in Section 4 below).

When the IP stack is required to use a source IP address of a specified type, it can use an existing address, or request a new IP prefix (of the same type) from the network and create a new one. If the host does not already have an IPv6 prefix of that specific type, it MUST request one from the network.

Using an existing address from an existing prefix is faster but might yield a less optimal route (if a hand-off event occurred after its configuration). On the other hand, acquiring a new IP prefix from the network may be slower due to signaling exchange with the network.

Applications can control the stack’s operation by setting a new flag - ON_NET flag - which directs the IP stack whether to use a preconfigured source IP address (if exists) or to request a new IPv6 prefix from the current serving network and configure a new IP address.

This new flag is added to the set of flags in the IPV6_ADDR_PREFERENCES option at the IPPROTO_IPV6 level. It is used in setsockopt() to set the desired behavior.

4. Usage example

4.1. Pseudo-code example

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

```c
#include <sys/socket.h>
```
# Socket information

```c
int s ; // socket id
```

// Source information (for setsc() and bind())

```c
sockaddr_in6 sourceInfo // my address and port for bind()
in6_addr sourceAddress // will contain the provisioned
// source IPv6 address
uint8_t sc_type = IPV6_REQUIRE_SESSION_LASTING_IP ;
// For requesting a Session-Lasting
// source IPv6 address
```

// Destination information (for connect())

```c
sockaddr_in6 serverInfo ; // server info for connect()
```

Create an IPv6 TCP socket

```c
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 IPv6 address before connecting
    if (setsc(s, &sourceAddress, &sc_type)) == 0){
        // Setting session continuity to Session Lasting is
        // Successful. sourceAddress now contains the Session-
        // Lasting source IPv6 address
    }
    // Bind to that source IPv6 address
    sourceInfo.sin6_family = AF_INET6 ;
    sourceInfo.sin6_port = 0 ; // let the stack choose the port
    sourceInfo.sin6_address = sourceAddress ;
    // Use the source address that was
    // generated by the setsc() call
    if (bind(s, &sourceInfo, sizeof(sourceInfo))==0){
        // Set the desired server’s information for connect()
        serverInfo.sin6_family = AF_INET6 ;
        serverInfo.sin6_port = SERVER_PORT_NUM ;
        serverAddress.sin6_addr = SERVER_IPV6_ADDRESS ;
    }
    // Connect to the server
    if (connect(s, &serverInfo, sizeof(serverInfo))==0) {
        // connect successful (3-way handshake has been
        // completed with Session-Lasting source address.
    }
}
// Continue application functionality
// ...
} // if connect() is successful
else {
  // connect failed
  // ...
  // Application code that handles connect failure and
  // closes the socket
  // ...
} // if connect() failed
} // if bind() successful
else {
  // bind() failed
  // ...
  // Application code that handles bind failure and
  // closes the socket
  // ...
} // if bind() failed
} // if setsc() was successful and of a Session-Lasting
// source IP address was provided
else {
  // application code that does not use Session-lasting IP
  // address. The application may either connect without
  // the desired Session-lasting service, or close the
  // socket...
} // if setsc() failed
} // if socket was created successfully

// The rest of the application’s code
// ...

4.2. Message Flow example

The following message flow illustrates a possible interaction for
achieving On-Demand functionality. It is an example of one scenario
and should not be regarded as the only scenario or the preferred one.

This flow describes the interaction between the following entities:

- Applications requiring different types of On-Demand service.
- The mobile host’s IP stack.
- The network infrastructure providing the services.
In this example, the network infrastructure provides 2 IPv6 prefixes upon attachment of the mobile host to the network: A Session-lasting IPv6 prefix and a Non-persistent IPv6 prefix. Whenever the mobile host moves to a different point-of-attachment, the network infrastructure provides a new Non-persistent IPv6 address.

In this example, the network infrastructure does not support Fixed IP addresses nor Graceful-replacement IP addresses.

Whenever an application opens an IP socket and requests a specific IPv6 address type, the IP stack will provide one from its available IPv6 prefixes or return an error message if the request cannot be fulfilled.

Message Flow:
- The mobile device attaches to the network.
- The Network provides two IPv6 prefixes: PREFsl1 - a Session-lasting IPv6 prefix and PREFnp1 - a Non-persistent IPv6 prefix.
- An application on the mobile host is launched. It opens an IP socket and requests a Non-persistent IPv6 address.
- The IP stack provides IPnp1 which is generated from PREFnp1.
- Another application is launched, requesting a Non-persistent IPv6 address.
- The IP stack provides IPnp1 again.
- A third application is launched. This time, it requires a Session-lasting IPv6 address.
- The IP stack provides IPSl1 which is generated from PREFsl1.
- The mobile hosts moves to a new point-of-attachment.
- The network provides a new Non-persistent IPv6 prefix - PREFnp2. PREFnp1 is no longer valid.
- The applications that were given IPnp1 re-establish the socket and receive a new IPv6 address - IPnp2 which is generated from PREFnp2.
- The application that is using IPSl1 can still use it since the network guaranteed that PREFsl1 will be valid even after moving to a new point-of-attachment.
- A new application is launched, this time requiring a Graceful-replacement IPv6 address.
- The IP stack returns setsc() with an error since the network does not support this service.
- The application re-attempts to open a socket, this time requesting a Session-lasting IPv6 address.
- The IP stack provides IPsl1.

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 On-Demand functionality will use the legacy API and will not be able to take advantage of the On-Demand Mobility feature.

Applications using the new On-Demand functionality should be aware that they may be executed in legacy environments that do not support it. Such environments may include a legacy IP stack on the mobile host, legacy network infrastructure, or both. In either case, the API will return an error code and the invoking applications may just give up and use legacy calls.

5.2. IP Stack in the Mobile Host

New IP stacks (that implement On Demand functionality) MUST continue to support all legacy operations. If an application does not use On-Demand functionality, the IP stack MUST respond in a legacy manner.

If the network infrastructure supports On-Demand functionality, the IP stack SHOULD follow the application request: If the application requests a specific address type, the stack SHOULD forward this request to the network. If the application does not request an address type, the IP stack MUST NOT request an address type and leave it to the network’s default behavior to choose the type of the allocated IP prefix. If an IP prefix was already allocated to the
host, the IP stack uses it and may not request a new one from the network.

5.3. Network Infrastructure

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

5.4. Merging this work with RFC5014

[RFC5014] defines new flags that may be used with setsockopt() to influence source IP address selection for a socket. The list of flags include: source home address, care-of address, temporary address, public address CGA (Cryptographically Created Address) and non-CGA. When applications require session continuity service and use setsc() and bind(), they SHOULD NOT set the flags specified in [RFC5014].

However, if an application erroneously performs a combination of (1) Use setsockopt() to set a specific option (using one of the flags specified in [RFC5014]) and (2) Selects a source IP address type using setsc() and bind(), the IP stack will fulfill the request specified by (2) and ignore the flags set by (1).

If bind() was not invoked after setsc() by the application, the IP address generated by setsc() will not be used and traffic generated by the socket will use a source IP address that complies with the options selected by setsockopt().

6. Summary of New Definitions

6.1. New APIs

setsc() enables applications to request a specific type of source IP address in terms of session continuity. Its definition is:
int setsc(int sockfd, in6_addr *sourceAddress, sc_type addressType);

Where:
- sockfd - is the socket descriptor of the socket with which a specific address type is associated
- sourceAddress - is a pointer to an area allocated for setsc() to place the generated source IP address of the desired session continuity type
- addressType - is the desired type of session continuity service. It is a 3-bit field containing one of the following values:
  0 - Reserved
  1 - FIXED_IPV6_ADDRESS
  2 - SESSION_LASTING_IPV6_ADDRESS
  3 - NON_PERSISTENT_IPV6_ADDRESS
  4 - GRACEFUL_REPLACEMENT_IPV6_ADDRESS
  5-7 - Reserved

setsc() returns the status of the operation:
- 0 - Address was successfully generated
- EAI_REQUIREDIPNOTSUPPORTED - the required service type is not supported
- EAI_REQUIREDIPFAILED - the network could not fulfill the desired request

setsc() MAY block the invoking thread if it triggers the TCP/IP stack to request a new IP prefix from the network to construct the desired source IP address. If an IP prefix with the desired session continuity features already exists (was previously allocated to the mobile host) and the stack is not required to request a new one as a result of setting the IPV6_REQUIRE_SRC_ON_NET flag (defined below), setsc() MAY return immediately with the constructed IP address and will not block the thread.

6.2. New Flags

The following flag is added to the list of flags in the IPV6_ADDR_PREFERENCE option at the IPPROTO6 level:

IPV6_REQUIRE_SRC_ON_NET - set IP stack address allocation behavior

If set, the IP stack will request a new IPv6 prefix of the desired type from the current serving network and configure a new source IP address. If reset, the IP stack will use a preconfigured one if it exists. If there is no preconfigured IP address of the desired type, a new prefix will be requested and used for creating the IP address.
7. Security Considerations

The different service types (session continuity types and address reachability) associated with the allocated IP address types, may be associated with different costs. The cost to the operator for enabling a type of service, and the cost to applications using a selected service. A malicious application may use these to generate extra billing of a mobile subscriber, and/or impose costly services on the mobile operator. When costly services are limited, malicious applications may exhaust them, preventing other applications on the same mobile host from being able to use them.

Mobile hosts that enables such service options, should provide capabilities for ensuring that only authorized applications can use the costly (or limited) service types.

The ability to select service types requires the exchange of the association of source IP prefixes and their corresponding service types, between the mobile host and mobile network. Exposing these associations may provide information to passive attackers even if the traffic that is used with these addressed is encrypted.

To avoid profiling an application according to the type of IP addresses, it is expected that prefixes provided by the mobile operator are associated to various type of addresses over time. As a result, the type of address could not be associated to the prefix, making application profiling based on the type of address harder.

The application or the OS should ensure that IP addresses regularly change to limit IP tracking by a passive observer. The application should regularly set the On Demand flag. The application should be able to ensure that session lasting IP addresses are regularly changed by setting a lifetime for example handled by the application. In addition, the application should consider the use of graceful replacement IP addresses.

Similarly, the OS may also associated IP addresses with a lifetime. Upon receiving a request for a given type of IP address, after some time, the OS should request a new address to the network even if it already has one IP address available with the requested type. This includes any type of IP address. IP addresses of type graceful replacement or non persistent should be regularly renewed by the OS.

The lifetime of an IP address may be expressed in number of seconds or in number of bytes sent through this IP address.
8. IANA Considerations

This document has no IANA considerations.

9. Contributors

This document was merged with [I-D.sijeon-dmm-use-cases-api-source]. We would like to acknowledge the contribution of the following people to that document as well:

Sergio Figueiredo
Altran Research, France
Email: sergio.figueiredo@altran.com

Younghan Kim
Soongsil University, Korea
Email: younghak@ssu.ac.kr

John Kaippallimalil
Huawei, USA
Email: john.kaippallimalil@huawei.com

10. Acknowledgements

We would like to thank Wu-chi Feng, Alexandru Petrescu, Jouni Korhonen, Sri Gundavelli, Dave Dolson Lorenzo Colitti and Daniel Migault for their valuable comments and suggestions on this work.

11. References

11.1. Normative References


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-07 (work
in progress), September 2017.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
A., Peterson, J., Sparks, R., Handley, M., and E.
Schooler, "SIP: Session Initiation Protocol", RFC 3261,
DOI 10.17487/RFC3261, June 2002,

[RFC5213] Gundavelli, S., Ed., Leung, K., Devarapalli, V.,
Chowdhury, K., and B. Patil, "Proxy Mobile IPv6",
RFC 5213, DOI 10.17487/RFC5213, August 2008,

[RFC5563] Leung, K., Dommety, G., Yegani, P., and K. Chowdhury,
"WiMAX Forum / 3GPP2 Proxy Mobile IPv4", RFC 5563,
DOI 10.17487/RFC5563, February 2010,

RFC 5944, DOI 10.17487/RFC5944, November 2010,


[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,

Korhonen, "Requirements for Distributed Mobility
Management", RFC 7333, DOI 10.17487/RFC7333, August 2014,

Authors' Addresses

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 November 6, 2015.

Copyright Notice

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

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

1. Introduction

In DMM scenario, mobility anchors would be deployed in a distributed manner, and as specified in RFC7333 [RFC7333], one of the aims of DMM is to reduce the routing redundancy between mobile node and correspondent node, which means providing a more optimal communication path for application traffic between mobile node and correspondent node. To achieve routing optimization for specific application traffic, the basic idea is to make the traffic using IP address(s) anchored at current anchor directly go to mobile node.

Different application sessions could require different IP address consistency support from network layer, for example, if the application layer cannot bear the IP address to be changed during the session, then the IP address for the session should be kept unchanged; but if the application layer could provide mechanism to deal with IP
address change, then the application could choose select a new optimal address during the session, to get reduce routing redundancy.

This document defines new IP network prefix types to assist address selection for applications.

2. Definition of New Prefix Types

Two types of IP network prefix are defined in this section. The extension of specific protocol to convey the defined new types of IP network prefix to MN is out of scope of this document.

Local home network prefix. It means that this prefix is allocated and advertised by current router which the MN attaches to. Remote home network prefix. It means that this prefix is allocated by another router instead of the router that the MN currently attaches to.

3. Mobile Node Operation

This section describes how the applications on the mobile node could select the optimal IP address based on different types of prefixes. For example, for on-going session, application always choose to use the prefix that it used before it handovers to a new location. For the newly initiate application, it will use the prefix that allocated by current router, e.g. local home network prefix. The mobile node can use advanced socket API to select the proper prefix, for example, extension to RFC 5014. The detail mechanism is out the scope of this document.

4. An Example of How This Draft Works

This section describes how the prefix types defined above solves the source address selection. Two different use cases are presented below. We assume a new Type flag "T" in Prefix Information option is used to indicate the new defined prefix type.

4.1. MN Handoffs From MAR to a Next MAR
T=L: Local home network prefix using common IP forwarding and routing mechanisms

T=R: Remote home network prefix using mobility anchoring and tunneling to maintain communications

Figure 1: Source address selection in DMM

As shown in figure 1, flow#1, flow#2 and flow#3 are initiated and anchored at MAR1, MAR2 and MAR3 respectively.

When MN attaches to MAR1, MAR1 sends Router Advertisement messages (RA) containing MN’s home network prefix (HNP1) in Prefix Information option with T=L. This indicates that HNP1 is local home network prefix which is allocated and advertised by current router (MAR1). MN can initiate a session with CN1 (i.e. flow#1 in figure 1) by using IPv6 addresses derived from HNP1. Common IPv6 routing mechanism will be applied for flow#1 as long as MN remains attached to MAR1.

When MN handoffs to MAR2 (flow#1 continues while MN handoffs), MAR2 sends RA messages containing MN’s new prefix (HNP2) in Prefix Information option with T=L together with old prefix (i.e. HNP1) with T=R. MN will learn that HNP2 is local home network prefix and HNP1 is remote home network prefix. At this moment, MN can initiate a new sessions with CN2 (i.e. flow#2 in the figure 1) by using IPv6 addresses derived from HNP2 as its source address. Because this IPv6 address is derived from a local home network prefix (i.e. HNP2), common IPv6 routing mechanism will be applied for flow#2. For flow#1, MAR1 plays role of LMA and MAR2 plays role of MAG. When MN handoffs to MAR3 (flow#1 and flow#2 continue while MN handoffs), MAR3 sends RA messages containing MN’s new prefix (HNP3) and previous prefixes (HNP1 and HNP2) in Prefix Information option with T=L for HNP3 and T=R for HNP1 and HNP2. This indicates HNP3 is local home network prefix, and HNP1 and HNP2 are remote home network prefixes. At this moment, MN can initiates new sessions with CN3 (i.e. flow#3 in figure 1) by using IPv6 addresses derived from HNP3 as its source address. And Common IPv6 routing mechanism will be applied for flow#3.

4.2. MN Handoffs From MAR Back to Its Previous MAR

MN could also handoff back from MAR3 to MAR2 again (assuming flow#1, flow#2 and flow#3 continue while MN handoffs).

In this case, as described above, MAR2 will send RA messages containing HNP1, HNP2 and HNP3 in Prefix Information option with T=L.
for HNP2 and T=R for HNP1 and HNP3. This indicates HNP2 is local home network prefix; HNP1 and HNP3 are remote home network prefixes.

Assuming that MN initiates a new sessions with a new communication node (e.g. with CN4 which is not shown in figure1) by using IPv6 addresses derived from HNP2 as its source address. Because this IPv6 address is derived from a local home network prefix (i.e. HNP2), common IPv6 routing mechanism will be applied for this session.

5. Security Considerations

TBD

6. IANA Considerations

This document makes no request of IANA.

7. References

7.1. Normative References


7.2. Informative References


[I-D.draft-seite-dmm-dma-00] Seite, P. and P. Bertin, "Distributed Mobility Anchoring,draft-seite-dmm-dma-00", February 2012.
Authors’ Addresses

Vic Liu
China Mobile
32 Xuanwumen West AVE, Xicheng, Beijing
Email: liuzhiheng@chinamobile.com

Hui Deng
China Mobile
32 Xuanwumen West AVE, Xicheng, Beijing
Email: denglingli@chinamobile.com

Dapeng Liu
Alibaba
Email: max@dotalks.com

Xinpeng Wei
Huawei
Email: Weixinpeng@huawei.com

Juan Liu
ZTE
No.68, Zijinhua RD, Yuhuatai District, Nanjing
Email: liu.juan45@zte.com.cn
Distributed mobility management deployment scenario and architecture
draft-liu-dmm-deployment-scenario-05

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 March 19, 2016.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in
Abstract

This document discusses the deployment scenario of distributed mobility management. The purpose of this document is to trigger the discussion in the group to understand the DMM deployment scenario and consideration from the operator’s perspective.

Table of Contents

Table of Contents ................................................ 2
1. Introduction .................................................. 2
2. Conventions used in this document.............................. 3
   2.1. Terminology .............................................. 3
3. Deployment Scenario and Model of DMM........................... 3
4. Network Function Virtualization Scenario....................... 4
   4.1. Network function virtualization deployment architecture... 4
   4.2. Control and data plane separation......................... 6
   4.3. Mobility management architecture.......................... 6
   4.4 NFV based deployment architecture......................... 7
5. SIPTO deployment scenario...................................... 8
6. WLAN deployment scenario....................................... 9
7. Conclusion ................................................... 10
8. Security Considerations....................................... 10
9. IANA Considerations .......................................... 10
10. Normative References......................................... 11
11. Informative References....................................... 11
12. Acknowledgments ............................................. 12
Authors’ Addresses .............................................. 12

1. Introduction

Distributed mobility management aims at solving the centralized mobility anchor problems of the traditional mobility management protocol. The benefit of DMM solution is that the data plane traffic does not need to traverse the centralized anchoring point. This document discusses the potential deployment scenario of DMM. The purpose of this document is to help the group to reach consensus
regarding the deployment model of DMM and then develop the DMM solution based on the deployment model.

2. Conventions used in this document

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].

2.1. Terminology

All the general mobility-related terms and their acronyms used in this document are to be interpreted as defined in the Mobile IPv6 base specification [RFC6275], in the Proxy mobile IPv6 specification [RFC5213], and in Mobility Related Terminology [RFC3753]. These terms include the following: mobile node (MN), correspondent node (CN), and home agent (HA) as per [RFC6275]; local mobility anchor (LMA) and mobile access gateway (MAG) as per [RFC5213], and context as per [RFC3753].

In addition, this draft introduces the following terms.

Location information (LI) function

is the logical function that manages and keeps track of the internet work location information of a mobile node which may change its IP address as it moves. The information may associate with each session identifier, the IP routing address of the MN, or of a node that can forward packets destined to the MN.

Forwarding management (FM)

is the logical function that intercepts packets to/from the IP address/prefix delegated to a mobile node and forwards them, based on internetwork location information, either directly towards their destination or to some other network element that knows how to forward the packets to their ultimate destination. With data plane and control plane separation, the forwarding management may be separated into a data-plane forwarding management (FM-DP) function and a control-plane forwarding management (FM-CP) function.

3. Deployment Scenario and Model of DMM

As discussed in the DMM requirement document, the centralized mobility management has several drawbacks. The main problem of the centralized mobility management protocols is that all the traffic need to anchor to a centralized anchor point. This approach does not
cause any problem in current mobile network deployment but in the scenario that will be discussed later in this document, centralized mobility management protocols will have many drawbacks and it is believed that DMM is more suitable in that scenario.

The main deployment scenario discussed in this document is divided into three scenarios. The first one is the network function virtualization scenario. In this scenario, the mobile core network’s control plane function is centralized in the mobile cloud. Apparently, deploying the data plane function also in the same centralized mobile cloud is not optimized from the traffic forwarding’s perspective. For the control plane The MME and PGW-F are implemented by NFV. For the dataplane the PGW-F/SGW-F can either be implemented by NFV or legacy devices. The second deployment scenario is the SIPTO/LIPA scenario which is discussed in 3GPP. In this scenario, DMM can provide optimized traffic offloading solution. The Third deploy scenario is the WLAN scenario. In this scenario, the AC is implemented in the cloud and the authentication status can maintained as the terminal move from one AP to another.

4. Network Function Virtualization Scenario

This section discusses network function virtualization scenario, the associated control - data plane separation and the possible mobility management functions to support this scenario.

4.1. Network function virtualization deployment architecture

The network function virtualization scenario is shown in Figure 1.
In this architecture, the mobile core include MME and PGW-F is located in the cloud data center, which can be the operator’s private cloud using NFV. The access network contain PGW-F/SGW-F is connected through an IP transit network. The PGW-F/SGW-F may also implement by NFV of small data center in convergence layer. The architecture of NFV based Mobile Core is shown in Figure 2.
In Figure 2, the MANO layer contains Orchestrator, VNFM and VIM. The Orchestrator is in charge of top-down service and source monitor and fulfillment. VNFM is in charge of managing the VNFs. And VIM normally is the Openstack which provides management of the whole virtualization layer.

4.2. Control and data plane separation

The cloud-based mobile core network architecture implies separation of the control and data planes. The control plane is located in the cloud and the data plane should be distributed. Otherwise, all the data traffic will go through the cloud which is obviously not optimized for mobile node to mobile node communication. For the mobile node to Internet communication, the Internet access point is normally located in the metro IP transit network. In this case, the mobile node to Internet traffic should also go through the Internet access point instead of the mobile core in the cloud.

However, in some deployment scenarios, the operator may choose to put the mobile core cloud in the convergence layer of IP metro network. In this case, the Internet access point may co-located with the mobile core cloud. In this case, the mobile node to Internet traffic may go through the mobile core cloud.

4.3. Mobility management architecture

Since the control plane and data plane are separated and the data plane is distributed, traditional mobility management cannot meet
this requirement. Distributed mobility management or SDN based mobility management may be used in this architecture to meet the traffic forwarding requirement (e.g. MN to MN and MN to Internet traffic should not go through from the mobile core cloud.). The traditional mobility management functions is not separating the data plane from the control plane. Basic mobility management functions include location information (LI) function and Forwarding management (FM). The former is a control plane function. The latter can be separated into data plane forwarding management (FM-DP) and control plane forwarding management (FM-CP).

The data plane function is FM-DP, while the control plane functions include FM-CP and LI. Then the control plane functions in the cloud-based mobile core includes LI and FM-CP. They are of cause other functions in the control plane such as policy function. The distributed data plane may have multiple instances of FM-DP in the network.

```
core network controller
    +--------+
    | LI, FM-CP |
    +--------+
    +--------+   +--------+   +--------+
    | FM-DP |   | FM-DP |   | FM-DP |
    +--------+   +--------+   +--------+

Figure 2: Mobility management functions with data plane - control plane separation under one controller
```

When the control of the access network is separate from that of the core, there will be separate controllers as shown in Figure 3.

```
Access network controller
    +--------+
    | LI, FM-CP |
    +--------+
    +--------+   +--------+   +--------+
    | FM-DP |   | FM-DP |   | FM-DP |
    +--------+   +--------+   +--------+

Core network controller
    +--------+
    | LI, FM-CP |
    +--------+
    +--------+   +--------+   +--------+
    | FM-DP |   | FM-DP |   | FM-DP |
    +--------+   +--------+   +--------+

Figure 3: Mobility management functions with separate controllers
```
Figure 2: Mobility management functions with data plane – control plane separation with separate control in core and in access networks.
4.4. NFV based deployment architecture

Here is the deployment architecture in NFV.

| [VNFO]              -------------------------------------------------- |
|  ------------    |  --------------------   ------------------------ ||VNFD(LI,FM-CP)(FD-DP)||
| |    NBI     |   | |  ----- ----- ----- | | ---------------------- |||
| | API Router|    |                  DataBase                        |
| |Core Engine|                                                        |

<table>
<thead>
<tr>
<th>[VNFM Driver]</th>
<th>[VIM Driver]</th>
<th>[PNF Driver]</th>
</tr>
</thead>
<tbody>
<tr>
<td>[VNFM]</td>
<td>[PNF]</td>
<td></td>
</tr>
<tr>
<td>VNF Life cycle management;</td>
<td></td>
<td>OVS</td>
</tr>
<tr>
<td>VNF configuration;</td>
<td></td>
<td>---------</td>
</tr>
<tr>
<td>VNF update;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VNF status monitor;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VNF Auto healing/Scale in/Scale out</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Vim |
|     |

| [VNF] LI Slicing ; FM-CP Slicing; FD-DP Slicing |

Figure 3 Deployment architecture
5. SIPTO deployment scenario

The Second deployment scenario is the SIPTO scenario which is discussed in 3GPP. DMM is believed to be able to provide dynamic anchoring. It allows the mobile node to have several anchoring points and to change the anchoring point according to the application requirement. In SIPTO scenario, the gateway function is located very near to the access network and to the user. If using current centralized mobility management, the traffic will need to tunnel back to the previous anchor point even when the mobile node has changed the point of attachment to a new one. Figure 3 shows the architecture of SIPTO.
6. WLAN deployment scenario

The Third deployment scenario is the WLAN scenario. DMM can enable the AC in the cloud. The cloud AC and maintain the authentication and connection status. As the terminal move from one AP to another, it still can have the connection.

............

('{{
  +--------------+
| Mobile AC     |
+--------------+
}}{{
  +++          +
|              |
  +++          +
}}{{
  '.............'
}}

Figure 4 SIPTO Scenario
7. Conclusion

This document discusses the deployment scenario of DMM. Three types of deployment scenario is discussed in this document. Further types of deployment scenario can be added to this document according to the progress of the group’s discussion.

8. Security Considerations

N/A

9. IANA Considerations

N/A
10. Normative References


11. Informative References


12. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.
Authors’ Addresses

Vic Liu
China Mobile
32 Xuanwumen West AVE, Xicheng, Beijing
Email: liuzhiheng@chinamobile.com

Dapeng Liu
Alibaba
Email: max@dotalks.com

H Anthony Chan
Huawei Technologies
5340 Legacy Dr. Building 3
Plano, TX 75024
USA
Email: h.a.chan@ieee.org

Hui Deng
China Mobile
32 Xuanwumen West AVE, Xicheng, Beijing
Email: denglingli@chinamobile.com

Xinpeng Wei
Huawei Technologies
Email: Xinpengwei@huawei.com
Abstract

In a network implementing Distributed Mobility Management, it has been agreed that Mobile Nodes (MNs) should exhibit agility in their use of IP addresses. For example, an MN might use an old address for ongoing socket connections but use a new, locally assigned address for new socket connections. Determining when to assign a new address, and when to release old addresses, is currently an open problem. Making an optimal decision about address assignment and release must involve a tradeoff in the amount of signaling used to allocate the new addresses, the amount of utility that applications are deriving from the use of a previously assigned address, and the cost of maintaining an address that was assigned at a previous point of attachment. As the MN moves farther and farther from the initial point where an address was assigned, more and more resources are used to redirect packets destined for that IP address to its current location. The MN currently does not know the amount of resources used as this depends on mobility path and internal routing topology of the network(s) which are known only to the network operator. This document provides a mechanism to communicate to the MN the cost of maintaining a given prefix at the MN’s current point of attachment so that the MN can make better decisions about when to release old addresses and assign new ones.

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."
1. Introduction

Previous discussions on address agility in distributed mobility management have focused on "coloring" prefixes with one of a small number of categories, such as Fixed, Sustained, or Nomadic. The assumption here is that the MN should use a permanent home address for sessions that need a persistent IP address, and a local, ephemeral address for short-lived sessions such as browsing. However, a small set of address categories lacks expressive power and leads to false promises being made to mobile nodes. For example, the concept that a home address can be maintained permanently and offered as an on-link prefix by any access router to which the MN may be attached in future is simply not attainable in the real world. There will always exist some access routers that do not have arrangements in place with the home network to re-route (via tunneling or other mechanisms) the home prefix to the current point of attachment.
Conversely, the assumption that a Nomadic prefix will never be available to an MN after it changes its current point of attachment is too limiting. There is no reason why an MN should not be able to keep a prefix that was assigned by a first network after it moves to a second network, provided that measures are put in place to re-route such prefixes to the new attachment point.

Rather, this document argues that there is in reality a continuum of cost associated with an address as the MN moves from one attachment point to another or from one network to another. The sources of the cost are the increased latency, network bandwidth, and network state being maintained by a network-based mobility management scheme to route packets destined to the prefix to the MN's current point of attachment. By communicating this cost to the MN every time its attachment point changes, the MN can make intelligent decisions about when to release old addresses and when to acquire new ones.

The cost should be communicated to the MN because of several constraints inherent in the problem:

1. The MN is the entity that must make decisions about allocating new addresses and releasing old ones. This is because only the MN has the information about which addresses are still in use by applications or have been registered with other entities such as DNS servers.

2. Only the network has information about the cost of maintaining the prefix in a network-based mobility management scheme, because the MN cannot know the network topology that gives rise to the inefficiencies.

If the cost of maintaining a prefix is not made available to the mobile node, it may attempt to infer the cost through heuristic mechanisms. For example, it can measure increased end-to-end latency after a mobility event, and attribute the increased latency to a longer end-to-end path. However, this method does not inform the MN about the network bandwidth being expended or network state being maintained on its behalf. Alternatively, a MN may attempt to count mobility events or run a timer in an attempt to guess at which older prefixes are more costly and in need of being released. However, these methods fail because the number of mobility events is not an indication of how far the MN has moved in a topological sense from its original attachment point which is what gives rise to the costs outlined above. Re-allocating an address upon expiration of a timer may introduce unnecessary and burdensome signaling load on the network and air interface.
1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

1.2. Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANDSF</td>
<td>Access Network Discovery and Selection Function</td>
</tr>
<tr>
<td>MN</td>
<td>Mobile Node</td>
</tr>
<tr>
<td>MPTCP</td>
<td>Multi-Path Transmission Control Protocol</td>
</tr>
<tr>
<td>ND</td>
<td>Neighbor Discovery</td>
</tr>
<tr>
<td>NGMN</td>
<td>Next Generation Mobile Networks</td>
</tr>
<tr>
<td>NUD</td>
<td>Neighbor Unreachability Detection</td>
</tr>
<tr>
<td>OMA-DM</td>
<td>Open Mobile Alliance - Device Management</td>
</tr>
<tr>
<td>PIO</td>
<td>Prefix Information Discovery</td>
</tr>
<tr>
<td>PGW</td>
<td>Packet data network Gateway</td>
</tr>
<tr>
<td>SeND</td>
<td>Secure Neighbor Discovery</td>
</tr>
<tr>
<td>SGW</td>
<td>Serving Gateway</td>
</tr>
</tbody>
</table>

2. Motivation

The Introduction speaks in general terms about the cost of a prefix. More specifically, we are talking about the aggregate amount of state being maintained in the network on behalf of the mobile node in addition to the transport resources being used (or wasted) to get packets to the MN’s current point of attachment.

In a non-mobile network, the addresses can be assigned statically in a manner that is aligned with the topology of the network. This means that prefix aggregation can be used for maximum efficiency in the state being maintained in such a network. Nodes deep in the network need only concern themselves with a small number of short prefixes, and only nodes near the end host need to know longer more specific prefixes. In the best case, only the last-hop router(s) need to know the actual address assigned to the end host. Also, routing protocols ensure that packets follow the least-cost path to the end host in terms of number of routing hops or according to other policies defined by the service provider, and these routing paths can change dynamically as links fail or come back into service.

However, mobile nodes in a wide-area wireless network are often handled very differently. A mobile node is usually assigned a fixed gateway somewhere in the network, either in a fixed central location or (better) in a location near where the MN first attaches to the network. For example, in a 3GPP network this gateway is a PGW that can be allocated in the home or visited networks. Initially, the cost of such a prefix is the state entry in the fixed gateway plus...
any state entries in intermediate tunneling nodes (like SGWs) plus whatever transport resources are being used to get the packet to the MN’s initial point of attachment.

When an MN changes its point of attachment, but keeps a fixed address, the cost of the prefix changes (usually it increases). Even if the fixed gateway was initially allocated very close to the initial point of attachment, as the MN moves away from this point, additional state must be inserted into the network and additional transport resources must be provided to get the packets to the current point of attachment. For example, a new SGW might be allocated in a new network, and now the packets must traverse the network to which the MN first attached before being forwarded to their destination, even though there may be a better and more direct route to communication peers from the new network. Whatever aggregation was possible at the initial point of attachment is now lost and tunnels must be constructed or holes must be punched in routing tables to ensure continued connectivity of the fixed IP address at the new point of attachment. Over time, as the MN moves farther and farther from its initial point of attachment, these costs can become large. When summed over millions of mobile nodes, the costs can be quite large.

Obviously, the assignment of a new address at a current point of attachment and release of the older, more costly prefix will help to reduce costs and may be the only way to meet emerging more stringent latency requirements [8]. However, the MN does not in general know the current cost of a prefix because it depends on the network topology and the number of handovers that have taken place and whether these handovers have caused the MN to transition between different topological parts of the network. It is the purpose of the protocol extension defined in this document to communicate the current cost of a prefix to the MN so that it can make intelligent decisions about when to get a new address and when to release older addresses. Only the MN can make a decision about when to release an address, because it is the only entity that knows whether applications are still listening waiting to receive packets at the old address.

Section 4 describes MN behavior when Router Advertisements with Prefix Cost is received.

3. Prefix Cost Sub-option

This document defines a prefix cost option to be carried in router advertisements. It is a sub-option that carries meta-data as defined by Korhonen et al. [7]
The prefix cost is carried as a 16-bit, unsigned number in network byte order. An higher number indicates an increased cost.

This sub-option is appended in Router Advertimsement messages that are sent on a periodic basis. No additional signaling cost is incurred to support this mechanism.

It should be noted that link layer events do not cause a change in the prefix cost.

The prefix cost is for a connection segment. No end-to-end congestion or flow control mechanisms are implied with this cost.

4. Host Considerations

Prefix Cost in a Router Advertisement PIO serves as a hint for the MN to use along with application knowledge, MN policy configuration on network cost and available alternative routes to determine the IP addresses and routes used. For example, if the application is downloading a large file, it may want to maintain an IP address and route until the download is complete. On the other hand, some applications may use multiple connections (e.g., with MPTCP) and may not want to maintain an IP address above a configured cost. It could also be the case that the MN maintains the IP address even at high cost if there is no alternative route/address. These decisions are made based on configured policy, and interaction with applications, all of which are decided by the MN.

When the MN is ready to release an IP address, it may send a DHCPv6 [5] Release message. The network may also monitor the status of a high cost connection with Neighbor Unreachability Detection (NUD) [2], [6], and determine that an address is not used after the NUD times out. The network should not continue to advertise this high cost route following the explicit release of the address or NUD timeout. It can initiate the release of network resources dedicated to providing the IP address to the MN.
The operator of the network or host’s service provider can configure policy that determines how the host should handle the prefix cost values. In a 3GPP network, the subscription provider may configure policies in the host via OMA-DM or S14 (ANDSF). For example, the service provider may configure rules to state that prefix cost values below 500 indicate low cost and ideal access network conditions, values from 501 - 5000 indicate that the host should try to relocate connections, and values above 5000 indicate a risk and impending loss of connectivity. The policies themselves can be (re-)configured as needed by the operator. Prefix cost information with each Router Advertisement allows the host to interpret a simple number and associated policies to (re-)select optimal routes. For networks service providers, when this cost is associated with charging, it can be a valuable tool in dynamically managing the utilization of network resources.

This draft does not aim to provide definitive guidance on how an OS or application process receives indications as a result of prefix cost option being conveyed in Router Advertisements. Only high level design options are listed here. New socket options or other APIs can be used to communicate the cost of an address in use on a given connection. For example, a new "prefix-cost" socket option, if set, can indicate that the application is interested in being notified when there is a change in the prefix cost. The actual mechanisms used to either notify or other means of busy polling on this change of prefix cost information need to be specified in other drafts. An alternative to the application discovering the changed prefix cost is to use a model where a connection manager handles the interface between the network and the application (e.g., Android Telephony Manager [9]). In this case, the connection manager is responsible to select and manage addresses based on policies (configured via OMA-DM or S14) and prefix cost obtained from the Router Advertisements.

5. Security Considerations

Security of the prefix cost option in the PIO needs to be considered. Neighbor Discovery (ND) and Prefix Information Option (PIO) security are described in [2] and [3]. A malicious node on a shared link can advertise a low cost route in the prefix cost option and cause the MN to switch. Alternatively, an incorrect higher cost route in the prefix cost option can result in the suboptimal use of network resources. In order to avoid such on-link attacks, SeND [4] can be used to reject Router Advertisements from nodes whose identities are not validated.
6. IANA Considerations

This memo defines a new Prefix Information Option (PIO) sub-option in Section 3.

7. References

7.1. Normative References


7.2. Informative References


Authors’ Addresses

Peter J. McCann (editor)
Huawei
400 Crossing Blvd, 2nd Floor
Bridgewater, NJ 08807
USA

Phone: +1 908 541 3563
Email: peter.mccann@huawei.com

John Kaippallimalil (editor)
Huawei
5340 Legacy Dr., Suite 175
Plano, TX 75024
USA

Email: john.kaippallimalil@huawei.com
DHCPv6 Extension for On Demand Mobility exposure
draft-moses-dmm-dhcp-ondemand-mobility-11

Abstract

Applications differ with respect to whether or not they need IP session continuity and/or IP address reachability. Networks providing the same type of service to any mobile host and any application running on the host yields inefficiencies. This document describes extensions to the DHCPv6 protocol to enable mobile hosts to indicate the required mobility service type associated with a requested IP prefix and to allow networks to indicate the type of mobility service associated with the allocated IP prefix in return.

Status of This Memo

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 https://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 1, 2019.

Copyright Notice

Copyright (c) 2019 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 (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect
1. Introduction

[I-D.ietf-dmm-ondemand-mobility] defines different types of mobility-associated services provided by access networks to mobile hosts with regards to maintaining IPv6 prefix continuity after an event of the host moving between locations with different points of attachments within the IP network topology. It further specifies means for applications to convey to the IP stack in the mobile host, their requirements regarding these services.

This document defines extensions to the DHCPv6 protocol ([RFC3315]) and [RFC3633] in the form of a new DHCP option that specifies the type of mobility services associated with an IPv6 prefix. The IP stack in a mobile host uses the DHCP client to communicate the type of mobility service it wishes to receive from the network. The DHCP server in the network uses this option to convey the type of service that is guaranteed with the assigned IPv6 prefix in return.

2. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119] [RFC8174] when, they appear in all capitals, as shown here.
3. IPv6 Continuity Service Option

The IPv6 Continuity Service option is used to specify the type of continuity service associated with a source IPv6 prefix. The IPv6 Continuity Service option MUST be encapsulated in the IA_prefix-options field of the IA_PD prefix option.

The format of the IPv6 Continuity Service option is:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| OPTION_IPv6_CONTINUITY_SERVICE |         option-length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| service-type  |
+-+-+-+-+-+-+-+-+
```

- **option-code**: OPTION_IPv6_CONTINUITYSERVICE (TBD)
- **option-len**: 1
- **service-type**: one of the following values:
  - Non-Persistent - a non-persistent IP prefix (1)
  - Session-Lasting - a session-lasting IP prefix (2)
  - Fixed - a fixed IP prefix (3)
  - Graceful-replacement - a graceful-replacement IP prefix (4)
  - Anytype - Anyone of the above (0)

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

All other values (5-255) are reserved for future use. If the OPTION_IPv6_CONTINUITYSERVICE option is received and its service-type is equal to one of the reserved values, the option SHOULD be ignored.
When a message is sent from a client to a server, the value of the IPv6 Continuity Service option indicates the type of continuity service required for the IPv6 prefix requested by the client.

When a message is sent from a server to a client, the value of the IPv6 Continuity Service option indicates the type of continuity service committed by the network for the associated IPv6 prefix. The value ‘AnyType’ SHOULD only appear in the message sent from the client to the server to indicate that the client has no specific preference. However, it cannot appear in a message sent from the server.

Once an IPv6 prefix type is requested and provided, any subsequent messages involving this prefix (lease renewal - for example) MUST include the IPv6 Continuity Service option with the same service type that was assigned by the server during the initial allocation.

If a server receives a request to assign an IPv6 prefix with a specified IPv6 Continuity service, but cannot fulfill the request, it MUST reply with the NoPrefixAvail status.

A server that does not support this option will ignore it and respond without taking into account the desired session continuity service. The response will not include the Continuity Service option encapsulated in the IAprefix-options field of the IA_PD prefix option.

The missing Continuity Service option in the response serves as an indication to the client that this feature is not supported by the server. It MAY use the allocated prefix knowing it does not necessarily support the desired Continuity service, or perform any other action.

A server MUST NOT include the IPv6 Continuity Service option in the IAprefix-options field of an IA_PD Prefix option, if not specifically requested previously by the client to which it is sending a message.

If a client receives an IA_PD Prefix option from a server with the IPv6 Continuity Service option in the IAprefix-options field, without initially requesting a specific service using this option, it MUST discard the received IPv6 prefix.

If the mobile device (host or router) has no preference regarding the type of continuity service it uses the ’AnyType’ value as the specified type of continuity service. The Server will allocate an IPv6 prefix with some continuity service and MUST specify the type in IPv6 Continuity Service option encapsulated in the IAprefix-options field.
field of the IA_PD Prefix option. The method for selecting the type of continuity service is outside the scope of this specification.

4. Correlation between Session Continuity Service and Lifetime Values

The values to be used in the Preferred-lifetime and Valid-lifetime fields in the IA Prefix Option are out of the scope of this specification and left to implementation. It is RECOMMENDED to provide longer lifetime values for Fixed and Session-lasting prefixes compared to the lifetime values of Non-persistent and Graceful-replacement prefixes because the network has guaranteed their validity regardless of the link to which the host is attached.

For clients using Graceful-replacement services, the network MAY obsolete a Prefix and allocate a new one from time to time especially in a mobility-related event. On such occasions, the network SHOULD provide a graceful period (lifetime) in which the obsoleted prefix can still be used and a new (longer) lifetime with the new prefix.

It is NOT RECOMMENDED using 0xFFFFFFFFFF (infinity) values for the lifetime of Fixed prefixes. Even though they are fixed, it is still safer to Rebind periodically. The lifetime value can be relatively long to reduce message exchange overhead.

Section 18.2 - Client Behavior of [I-D.ietf-dhc-rfc3315bis] specifies that when a client detects that it may have moved to a new link, it uses Rebind if it has delegated prefixes. It is worth clarifying that a client does not HAVE to Rebind the prefixes if they are Fixed or Session-lasting prefixes.

5. Security Considerations

There are no specific security considerations for this option.

6. IANA Considerations

TBD

7. References

7.1. Normative References

7.2. Informative References

[I-D.ietf-dhc-rfc3315bis]
Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A.,
Richardson, M., Jiang, S., Lemon, T., and T. Winters,
"Dynamic Host Configuration Protocol for IPv6 (DHCPv6)
bis", draft-ietf-dhc-rfc3315bis-13 (work in progress),
April 2018.

[I-D.ietf-dmm-distributed-mobility-anchoring]
Chan, A., Wei, X., Lee, J., Jeon, S., and C. Bernardos,
"Distributed Mobility Anchoring", draft-ietf-dmm-
distributed-mobility-anchoring-11 (work in progress),
August 2018.

[I-D.ietf-dmm-ondemand-mobility]
Jeon, "On Demand Mobility Management", draft-ietf-dmm-

C., and M. Carney, "Dynamic Host Configuration Protocol
for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July

[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,

[RFC7934]  Colitti, L., Cerf, V., Cheshire, S., and D. Schinazi,
"Host Address Availability Recommendations", BCP 204,
RFC 7934, DOI 10.17487/RFC7934, July 2016,

Authors' Addresses

Danny Moses
Intel
Petah Tikva
Israel

Email: danny.moses@intel.com
Distributed Mobility Management Protocol for WiFi Users in Fixed Network
draft-sarikaya-dmm-for-wifi-05.txt

Abstract

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

Status of This Memo

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 https://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 May 3, 2018.

Copyright Notice

Copyright (c) 2017 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 (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect
1. Introduction

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

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

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

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

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

3. Overview

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

Access routers (AR) are Unified Gateways (UGW) that are the access network gateways that behave similarly as Evolved Packet Core (EPC) Edge Router (EPC-E) in [I-D.matsushima-stateless-uplane-vepc]. UGW is configured an anycast address on the interface facing the Residential Gateway (RG). RGs use this address to forward packets from the users. The fixed access network delivers the packets to geographically closest UGW. UGW plays the role of Access Data Plane Node (A-DPN) defined in [I-D.ietf-dmm-deployment-models]. A-DPN and UGW are interchangeably used in this document.
Wi-Fi smart phone, the mobile node (MN) is assigned a unique prefix using either Stateless Address Auto Configuration (SLAAC) or by a DHCP server which could be placed in the cloud. In case of SLAAC, RG is delegated the prefixes by DHCP server using [RFC3633].

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

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

When MN moves from one A-DPN into another A-DPN, IP mobility signaling needs to be introduced. In this document we use Handover Initiate/Handover Acknowledge (HI/HACK) messages defined in [RFC5949]. Handover Initiate message can be initiated by either previous UGW (predictive handover) or the next UGW (reactive UGW).
In reactive handover, RG establishes a new connection with the next UGW when MN moves to this RG and provides previous UGW address. This will trigger the next UGW to send HI message to the previous UGW. Previous UGW sends HACK messages which establishes a tunnel between previous and next UGWs. Previous UGW sends packets destined to MN to the new UGW which in turn sends them to MN.

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

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

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

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

4. Detailed Protocol Operation

In this section, Layer 2 and Layer 3 mobility procedures are explained.
4.1. Layer 2 Mobility in Access Network

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

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

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

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

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

4.2. Layer 3 Mobility and Routing in Core Network

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

Predictive handover happens when MN just before leaving the previous RG (pRG) for the next RG (nRG) MN is able to send an 802.11 message
containing MN MAC address and nRG MAC address, e.g. learned from beacons to the pRG (called Leave Report in Figure 2. pRG then sends a handover indication message to pUGW providing MN and nRG addresses (called Leave Indication) and this could happen between two respective virtual machines in the cloud. This message results in pUGW getting nUGW information and then sending Handover Initiate message to nUGW, which also could happen in the cloud. nUGW replies with Handover Acknowledge message. pUGW sends any packets destined to MN to nUGW after being alerted by the control plane. MN moves to nRG and nUGW is informed about this from Layer 2 mobility

<table>
<thead>
<tr>
<th>MN</th>
<th>P-RG</th>
<th>N-RG</th>
<th>(P-UGW)</th>
<th>(N-UGW)</th>
<th>Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>Leave Report</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b)</td>
<td></td>
<td>Leave</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(c)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(d)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: Predictive Handover

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

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

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

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

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

Figure 3: Reactive Handover

Note that Handover Initiate and Handover Acknowledge messages used in this document carry only a subset of parameters defined in [RFC5949]. Also no involvement with the Local Mobility Anchor (LMA) [RFC5213] is needed.
be populated with at least one entry in the device, and additional entries may be configured by a client. Normally the server creates the required item as an entry in state data. Additional entries may be created in the configuration by a client via the NETCONF protocol using RPC messages like edit-config and copy-config.

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

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

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

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

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

Sarikaya & Xue Expires May 3, 2018 [Page 9]
Route propagation at the lowest level starts with I2RS Agent as NETCONF Server in nUGW informing the handover to I2RS Client as NETCONF Client upstream. I2RS Client then checks any routing information for MN by first using get-config to retrieve the active route for MN to make sure that none exits and MN prefix is topologically incorrect. Next I2RS client issues an edit-config message with create operation to add a host route for the new MN. I2RS Client then informs this route to I2RS Client upstream which creates a similar route at the I2RS Agent upstream.

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

4.4. Authentication

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

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

- Preauthentication: When The MN supplicant may authenticate with both pRG and nRG at a time. Successful completion of EAP authentication between the MN and nRG establishes a pair of PMKSA on both the MN and nRG. When the MN moves to the nRG, the authentication has already done, which is shown as follows.
o Cached PMK: The RG reserves the PMK as a result of previous authentication. When the MN is roaming back to the previous RG, if a successful EAP authentication has happened. The MN can retain the 802.11 connection based on PMK information reserved. When the authentication is handled by the UGW as an Authenticator. When the MN moves to the nRG, a join report packet will be initiated from the MN to nRG for IEEE802.11 connection to the same UGW. The nRG can retain the PMK information from the UGW which is reserved during the successful authentication procedure between the MN and the pRG, as shown in Figure 4.
When a MN moves at home or in a hot spot from one AP to another AP in the same UGW, it is possible that it may undergo a full EAP authentication (as defined in [RFC3748]). However, there are several simple authentication methods (defined in [IEEE-802.11-2007]):

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

- Preauthentication: MN supplicant may authenticate with both pRG and nRG at a time. Successful completion of EAP authentication between the MN and nRG establishes a pair of PMKSA on both the MN and nRG. When the MN moves to the nRG, the authentication has already been completed, which is shown as follows.
o Cached PMK: The RG reserves the PMK as a result of previous authentication. When the MN is roaming back to the previous RG, if a successful EAP authentication has happened. The MN can retain the 802.11 connection based on PMK information reserved. When the authentication is handled by the UGW as an Authenticator. When the MN moves to the nRG, a join report packet will be initiated from the MN to nRG for IEEE802.11 connection to nUGW. The nRG can retain the PMK information from the nUGW, the nUGW may can retain the reserved PMK from the pUGW based on HI message.
The above Layer 2 operations do not affect Layer 3. MN does not change the prefix assigned to it initially.

Note that charging solution is not described in this version.

5. Multicast Support

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

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

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

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

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

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

5.1. IPv4 Support

IPv4 can be supported similarly as in vEPC [I-D.matsushima-stateless-uplane-vepc]. UGW stays as IPv6 node receiving from all ROs IPv6 packets and forwarding them upstream.
IPv4 MN is supported at the RG. RG has B4 functionality of DS-Lite [RFC6333], CLAT entity for 464XLAT [RFC6877], Lightweight B4 [RFC7596] or MAP Customer Edge [RFC7597]. RG encapsulates IPv4 packets using these protocols into IPv6 packets making sure that UGW stays IPv6 only.

6. IANA Considerations

TBD.

7. Security Considerations

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

8. Acknowledgements

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

9. References

9.1. Normative References

[I-D.ietf-dmm-deployment-models]

[IEEE-802.11-2007]

[IEEE-802.11i]


Internet-Draft            DMM for Wifi Use Case             October 2017


9.2. Informative References


Appendix A. YANG and RPC Programs

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

A.1. Host Routing Module

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

A.2. Route Establishment RPCs

This program runs on ietf-ipv6-unicast-host-routing YANG module which has been obtained from ietf-ipv6-unicast-routing module by defining the hostroute as a list of host routes. First issue a get-config on the configuration data to extract the existing route for the host whose prefix is destination-prefix and the next-hop is the next-hop address. Delete the route at pUGW. This procedure deletes the route at pUGW.
Add a new route for MN at nUGW. This route is based on MN’s prefix, destination-prefix and the upstream router to which MN’s traffic should routed, next-hop-address.

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

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

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

This RPC procedure shows a get-config filter to find a record in the routing information base for a specific host whose prefix is 2001:db8:1:0::/64 and the next-hop is 2001:db8:0:1::2. It could be used for the get-config’s in Appendix A.2. We validated this procedure using the public domain tool pyang.
<rpc message-id="101"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
 xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces"
 xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <t:top xmlns:t="urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-host-routing">
        <t:routing-instance>t:rtr0</t:routing-instance>
        <t:rib>
          <t:routes>
            <t:route>
              <t:destination-prefix>2001:db8:1:0::/64</t:destination-prefix>
              <t:outgoing-interface>eth1</t:outgoing-interface>
              <t:next-hop-address>2001:db8:0:1::2</t:next-hop-address>
            </t:route>
            </t:routes>
          </t:rib>
        </t:top>
      </filter>
    </get-config>
  </rpc>

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

This RPC procedure shows an edit-config procedure to create a new host route in the routing information base for a specific host whose prefix is 2001:db8:1:0::/64 and the next-hop is 2001:db8:0:1::2. It could be used for the edit-config’s in Appendix A.2. We validated this procedure using the public domain tool pyang.
<rpc message-id="101"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:if="urn:ietf:params:xml:ns:yang:ietf-interfaces"
    xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
    <edit-config>
        <target>
            <running/>
        </target>
        <default-operation>none</default-operation>
        <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
            <top xmlns="urn:ietf:params:xml:ns:yang:ietf-ipv6-unicast-host-routing">
                <routing-instance> rtr0 </routing-instance>
                <rib>
                    <routes>
                        <route xc:operation="create">
                            <destination-prefix>2001:db8:1:0::/64</destination-prefix>
                            <outgoing-interface>eth1</outgoing-interface>
                            <next-hop-address>2001:db8:0:1::2</next-hop-address>
                        </route>
                    </routes>
                </rib>
            </top>
        </config>
    </edit-config>
</rpc>

Authors' Addresses

Behcet Sarikaya

Email: sarikaya@ieee.org

Li

Email: xueliucb@gmail.com
Abstract

This document presents available deployment models for distributed mobility management networks, consisted of mobility management functions: anchoring function, location management, and forwarding management functions defined in RFC7429. Some of the functions are modified on a need to allow potential deployment scenarios support.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 8, 2017.

Copyright Notice

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
1. Introduction

This draft presents available deployment models consisted of mobility management functions defined in [RFC7429], for distributed mobility management (DMM) networks. With the mobility management functions in [RFC7429], i.e. anchor function (AF), location management function (LM), and forwarding management function (FM), centralized mobility management solutions such as Mobile IP (MIP), Hierarchical Mobile IPv6 (HMIPv6), and Proxy Mobile IPv6 (PMIPv6) have been described and decomposed by functional aspects, trying to analyze gaps from the requirements for DMM [RFC7333]. In this draft, with the functions, we sketch and describe the deployment models for DMM networks, accommodating the possible DMM solutions as well as providing an insight to understand the potentials of DMM. We also describe where the presented deployment models are substantiated with solution proposals submitted in DMM WG.

2. Conventions and Terminology

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

Following terms come from [RFC7429] with modified definition in the AF.
Anchorig Function (AF) is defined as a combined control-plane and data-plane functions. For the control-plane function, it allocates an IP address, i.e., Home Address (HoA), or prefix, i.e., Home Network Prefix (HNP) a mobile node, topologically anchored by the advertising node. That is, the anchor node is able to advertise a connected route into the routing infrastructure for the allocated IP prefixes. It also takes a data-plane anchor point where packets destined to the IP address or IP prefix allocated by the anchor should pass through.

The AF can be deployed in a decoupled way, i.e. separated control plane and data plane. In that case, following two terms - AF Control Plane (AF-CP) and AF Data Plane (AF-DP) - are used. AF-CP is responsible of allocating the IP address and advertising a connected route for an associated terminal while AF-DP is responsible of anchoring received data packets destined to the IP address allocated by the anchor.

Internetwork Location Management (LM) is a control-plane function, which manages and keeps track of the internetwork location of an MN. The location information may be a binding of the advertised IP address/prefix, e.g., HoA or HNP, to the IP routing address of the MN, or it may be a binding of a node that can forward packets destined to the MN. Note that the LM could belong to the AF-CP, as it is done in several solutions, i.e. Mobile IP (MIP) and Proxy Mobile IPv6 (PMIPv6). However, in this draft, each function is indicated distinctively, as those functions could be deployed in different locations to allow advanced control and smooth evolution for DMM.

Forwarding Management (FM) function performs 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. Following the FM definition in [RFC7429], it may be split into the control plane (FM-CP) and data plane (FM-DP).

3. Deployment Models

We specify and analyze expected use cases where the MN tries to initiate an application.

3.1. D1: Distributed AM, LM, and FM (with centralized LM) - All-in-One
Figure 1. Distributed AM, LM, and FM functions (with centralized LM)

In this deployment model, AF, LM, and FM functions are co-located in every mobility router deployed at edge. This model can be called All-in-One for DMM. Depending on the use of the central LM, the model can be distinguished into fully distributed or partially distributed. In the partially distributed case, interface (a), between the centralized LM and the mobility routers shown in Fig. 1, is could be used for querying necessary mapping information by the edge mobility routers. Interface (b), between the mobility routers shown in Fig. 1, is used for conveying control signaling messages to control a forwarding path between them. Solutions following the given model could be [I-D.seite-dmm-dma][I-D.bernardos-dmm-pmip].

3.2. D2: Distributed AF-DP, LM and FM with centralized AF-CP (+ LM)
In this model, we distinguish AF with AF Control Plane (AF-CP) and AF Data Plane (AF-DP). AF-DP is distributed with LM and FM into deployed mobility routers while AF-CP is centralized in a single entity, following a trend of separation of control and user plane for mobility management. For an extensive scenario support, LM may be co-located with the AF-CP. AF-DP is determined by the AF-CP. One possible solution could be to use such as User-Plane Address option to deliver AF-DP IP address serving router or terminal should contact, as proposed in [RFC7389]. Interface (a) shown in Fig. 2 is used to control AF-DP function, with signaling messages or configuration information. Interface (b) shown in Fig. 2 is used for establish and control the forwarding path between the mobility routers.

3.3. D3: Distributed AF-DP and FM-DP with centralized AF-CP, LM, and FM-CP
In the model, separation of FM Control Plane (FM-CP) and FM Data Plane (FM-DP) is applied with the separation of AF-CP and AF-DP. The LM is located at the central entity. Comparing D3 with D2, D3 can provide smooth and flexible forwarding path management between the AF-DP of an allocated IP address and the current serving router where the terminal is attached. Interface (a) shown in Fig. 3 is used to control AF-DP and FM-DP function by the respective control functions, AF-CP and FM-CP, with signaling messages or configuration information. [I-D.ietf-dmm-fpc-cpdp] presents a framework that can facilitate forwarding policy configuration, based on D3 model, imparting a role and characteristics of a mobility router as well as configuring a forwarding path. [I-D.matsushima-stateless-uplane-vepc] may be subject to D3 model, the control functions in vEPC delivers Route Update to EPC Edge Routers, to configure a data-plane routing path.

4. IANA Considerations

This document makes no request of IANA.

5. Security Considerations

T.B.D.
6. Acknowledgements

7. References

7.1. Normative References

[I-D.ietf-dmm-fpc-cpdp]  


7.2. Informative References

[I-D.bernardos-dmm-pmip]  

[I-D.matsushima-stateless-uplane-vepc]  

[I-D.seite-dmm-dma]  


Authors’ Addresses

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

Email: seiljeon@skku.edu

Younghan Kim
Soongsil University
369, Sangdo-ro, Dongjak-gu
Seoul 156-743
Korea

Email: younghak@ssu.ac.kr
Abstract

In the basic Proxy Mobile IPv6 (PMIPv6) specification, a Mobile Node (MN) is assigned with a 64-bit Home Network Prefix (HNP) during its initial attachment for the Home Address (HoA) configuration. During the movement of the MN, this prefix remains unchanged and in this way it is unnecessary for the MN to reconfigure its HoA and reconnect the ongoing communications. However, the current protocol (RFC5213) does not specify related operations to support the MN to timely receive and use a new HNP when the allocated HNP changes. In this draft, a possible solution to support the HNP renumbering is proposed, as an update of RFC5213.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

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 October 14, 2015.
1. Introduction

Network managers currently prefer to Provider Independent (PI) addressing for IPv6 to attempt to minimize the need for future possible renumbering. However, widespread use of PI addresses may create very serious Border Gateway Protocol (BGP) scaling problems. It is thus desirable to develop tools and practices that may make IPv6 renumbering a simpler process to reduce demand for IPv6 PI space [RFC6879]. In this draft, we aims to solve the HNP renumbering problem when the HNP in PMIPv6 [RFC5213] is not a PI type.

2. Usage scenarios

There are a number of reasons why the HNP renumbering support is a useful. A few are identified below:

- Scenario 1: the PMIPv6 service provider is assigned with the HNP set from the (uplink) Internet Service Provider (ISP), and then the HNP renumbering may happen if the PMIPv6 service provider switches to a different ISP.
o Scenario 2: multiple Local Mobility Anchors (LMAs) may be deployed by the same PMIPv6 service provider, and then each LMA may serve for a specific HNP set. In this case, the HNP of an MN may change if the current serving LMA switches to another LMA but without inheriting the assigned HNP set [RFC6463].

o Scenario 3: the PMIPv6 HNP renumbering may be caused by the re-building of the network architecture as the companies split, merge, grow, relocate or reorganize. For example, the PMIPv6 service provider may reorganize its network topology.

In the scenario 1, we assume that only the HNP is renumbered while the serving LMA remains unchanged and this is the basic scenario of this document. In the scenario 2 and 3, more complex results may be caused, for example, the HNP renumbering may happen due to the switchover of serving LMA.

In the Mobile IPv6 (MIPv6), when the home network prefix changes (maybe due to the above reasons), the Home Agent (HA) will actively notify the new prefix to the MN and then the renumbering of the HoA can be well supported [RFC6275]. While in the basic PMIPv6, the PMIPv6 binding is triggered by the Mobile Access Gateway (MAG), which detects the attachment of the MN. When the HNP renumbering happens, a scheme is also needed for the LMA to immediately initiate the PMIPv6 binding state refreshment. Although this issue is also discussed in the [RFC5213] (Section 6.12), the related solution has not been specified.

3. Protocol

When the HNP renumbering happens in PMIPv6, the LMA has to notify the new HNP to the MAG that has to announce the new HNP to the MN accordingly. Also, the LMA and the MAG must delete the created routing states for the renumbered prefix. To support this procedure, [RFC7077]can be adopted which specifies asynchronously update from the LMA to the MAG about the updated session parameters. This document considers the following two cases:

(1) HNP is renumbered in the same LMA

In this case, the LMA remains unchanged as in the scenario 1 and scenario 3. The operation steps are shown in Figure 1.
When the PMIPv6 service provider renumbers the HNP set in the same LMA, the serving LMA will initiate the HNP renumbering operation. The LMA allocates a new HNP for the related MN.

The LMA sends the Update Notification (UPN) message to the MAG to update the HNP information. If the DHCP is used in PMIPv6 to allocate the HoA, the new HNP should be also notified to the DHCP infrastructure.

After the MAG receives this UPN message, it recognizes that the related MN has a new HNP. Then the MAG should notify the MN about the new HNP with an RA message or allocate a new address within the new HNP with a DHCP message.

When the MN obtains the new HNP information, it deletes the old HoA and configures a new HoA (with the newly allocated HNP).

The MAG sends back the Update Notification Acknowledgement (UNA) to the LMA for the notification of successful update of the HNP, related binding state, and routing state. Then the LMA updates the routing information corresponding to the MN to replace the old HNP with the new one.

(2) HNP renumbering caused by LMA switchover

Because the HNP is assigned by the LMA, the HNP renumbering may be caused by the LMA switchover, as in the scenario 2 and scenario 3.
The information of LMA is the basic configuration information of MAG. When the LMA changes, the related profile should be updated by the operator. In this way, the MAG will initiate the re-registration to the new LMA as specified in RFC5213. When the HNP renumbering is caused in this case, the new HNP information will be sent by the LMA. Accordingly, the MAG will withdraw the old HNP information of the MN and advise the new HNP to the MN as related steps in Section 3.1.

4. Message format

(1)UPN message

In the UPN message sent from the LMA to the MAG, the notification reason is set to 2 (UPDATE-SESSION-PARAMETERS). Besides, the HNP Option containing the new HNP and the Mobile Node Identifier Option carrying Identifier of MN are contained as Mobility Options of UPN.

(2)RA Message

When the RA message is used by the MAG to advise the new HNP, two Prefix Information options are contained in the RA message [RFC2461]. In the first Prefix Information Option, the old HNP is carried but both the related Valid Lifetime and Preferred Lifetime are set to 0. In the second Prefix Information Option, the new HNP is carried with the Valid Lifetime and Preferred Lifetime set to larger than 0.

(3)DHCP Message

When the DHCP is used in PMIPv6 to configure the address for the MN, a new IPv6 HoA is generated based on the new HNP. Trigged by the UPN message, the MAG will request the new HoA from the DHCP server first and then the MAG updates the allocated HoA to the MN through the DHCP server-initiated configuration exchange [RFC3315].

5. Other issues

In order to maintain the reachability of the MN, the DNS resource record corresponding to this MN may need to be updated when the HNP of MN changes [RFC3007]. However, this is out the scope of this draft.

6. Security considerations

This extension causes no further security problem. The security considerations in [RFC5213] and [RFC7077] are enough for the basic operation of this draft.

Other security issues will be analyzed further.
7. Normative References


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

Xiaodong Lee  
CNNIC  
No.4 South 4th Street, Zhongguancun  
Beijing  100190  
China  
EMail: xl@cnnic.cn