INTERNET-DRAFT
Intended Status: Informational
Expires: September 2018

K. Bogineni
Verizon
A. Akhavain
Huawei Technologies Canada
T. Herbert
Quantonium
D. Farinacci
lispers.net
A. Rodriguez-Natal
Cisco
March 5, 2018

Optimized Mobile User Plane Solutions for 5G
draft-bogineni-dmm-optimized-mobile-user-plane-00.txt

Abstract

3GPP CT4 has approved a study item to study different mobility management protocols for potential replacement of GTP tunnels between UPFs (N9 Interface) in the 3GPP 5G system architecture.

This document provides an overview of 5G system architecture in the context of N9 Interface which is the scope of the 3GPP CT4 study item [23501, 23502, 23503, 23203, 29244, 29281, 38300, 38401]. The requirements for the network functions and the relevant interfaces are provided.

Reference scenarios and criteria for evaluation of various IETF protocols are provided.

Several IETF protocols are considered for comparison: SRv6, LISP, ILA and several combinations of control plane and user plane protocols.

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 September 6, 2018.

Copyright Notice

Table of Contents

1 Introduction and Problem Statement

3GPP CT4 WG has approved a study item [CT4SID] to study user-plane protocol for N9 in 5GC architecture as specified in TS 23.501 [23501] and TS 23.502 [23502] for Rel-15. This provides an opportunity to investigate potential limits of the existing user plane solution and potential benefits of alternative user plane solutions.

IETF has some protocols for potential consideration as candidates. These protocols have the potential to simplify the architecture through reduction/elimination of encapsulation; use of native routing mechanisms; support of anchor-less mobility management; reduction of session state and reduction of signaling associated with mobility management.

This document comprehensively describes the various protocols and how they can be used in the 3GPP 5G architecture. Specifically Segment Routing v6 (SRv6), Locator Identifier Separation Protocol (LISP) and Identifier Locator Addressing (ILA) are described in the context of the 3GPP 5G architecture for several scenarios: as a replacement of GTP on N9; as a replacement of GTP in the whole system; integrated with transport; used in specific network slices, etc.

A comparison of the various protocols is also provided.

2 Conventions used in this document

In examples, "C:" and "S:" indicate lines sent by the client and server respectively.

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].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying significance described in RFC 2119.

In this document, the characters ">>" preceding an indented line(s) indicates a statement using the key words listed above. This convention aids reviewers in quickly identifying or finding the portions of this RFC covered by these keywords.

3 Overview of Existing Architecture and Protocol Stack

This section briefly describes the 5G system architecture as specified in 3GPP TS 23.501. The key relevant features for session management and mobility management are:

- Separate the User Plane (UP) functions from the Control Plane (CP) functions, allowing independent scalability, evolution and flexible deployments e.g. centralized location or distributed (remote) location.

- Support concurrent access to local and centralized services. To support low latency services and access to local data networks, UP functions can be deployed close to the Access Network.

- Support roaming with both Home routed traffic as well as Local breakout traffic in the visited PLMN.

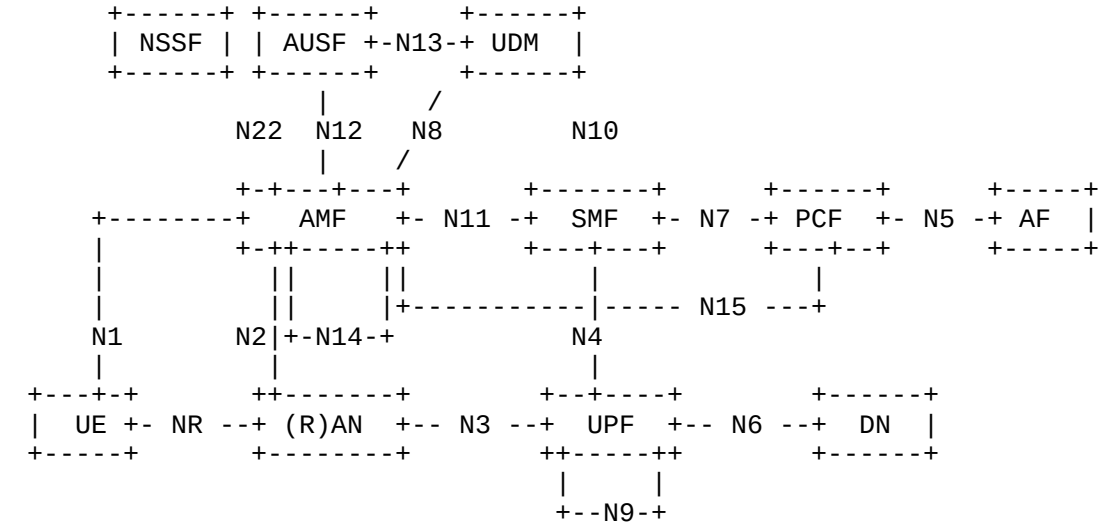


Figure 1: 5G System Architecture in Reference Point Representation

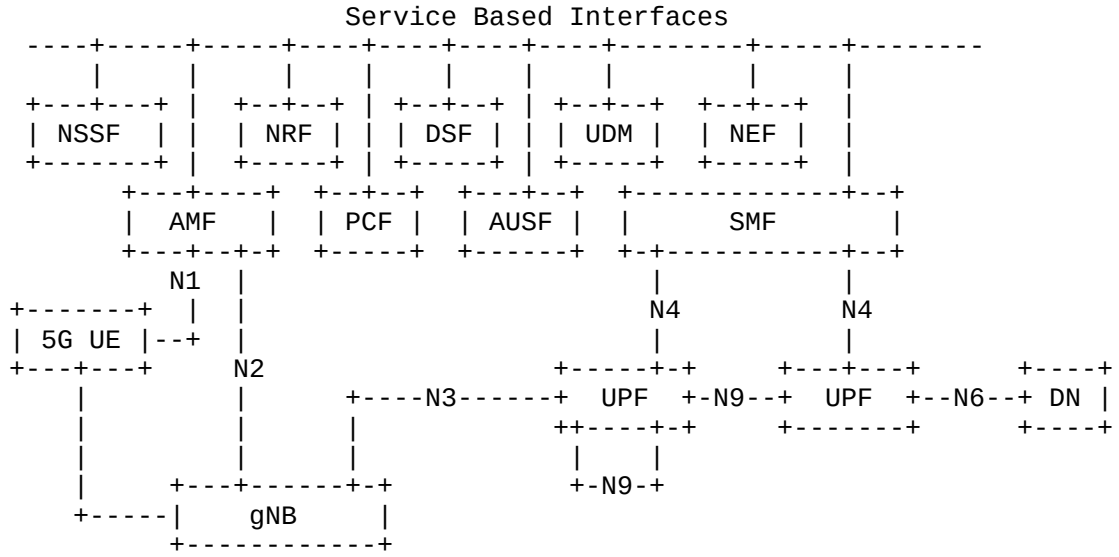


Figure 2: 5G Services Based Architecture

The roaming architectures are depicted in the two figures below.

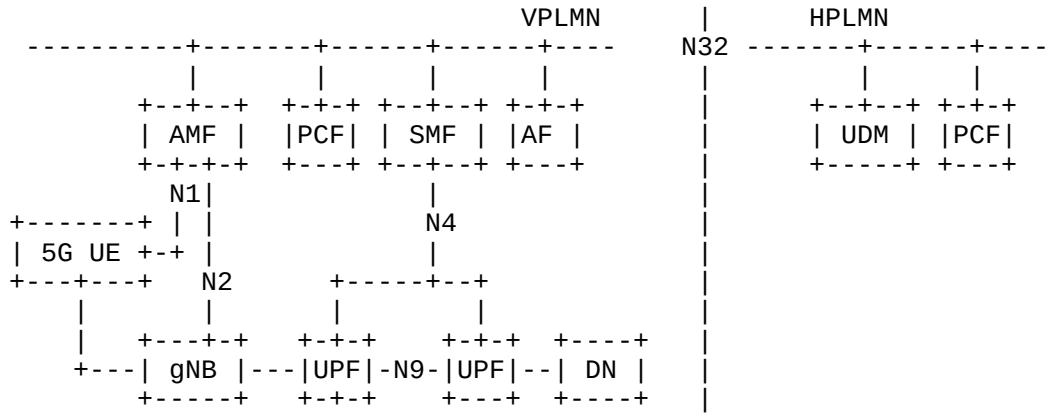


Figure 3: Roaming 5G System architecture- local breakout scenario

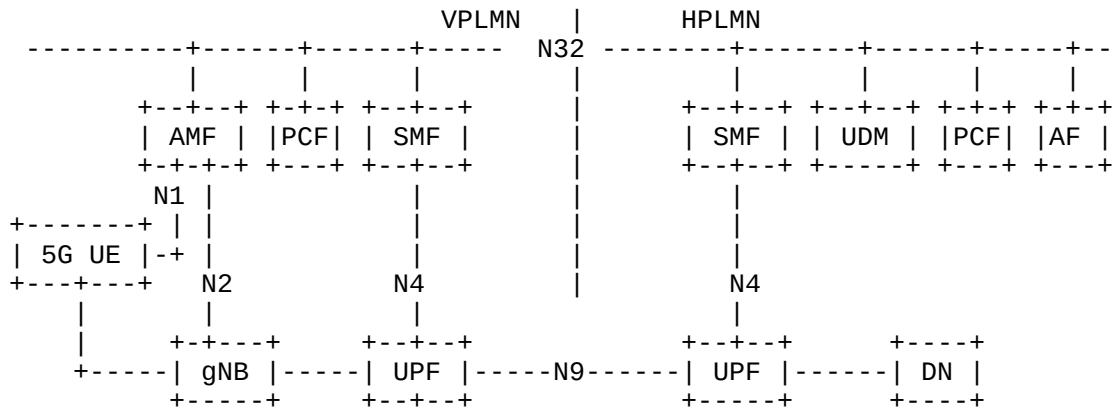


Figure 4: Roaming 5G System architecture - home routed scenario

Figure 5 depicts the non-roaming architecture for UEs concurrently accessing two (e.g. local and central) data networks using multiple PDU Sessions, using the reference point representation. This figure shows the architecture for multiple PDU Sessions where two SMFs are selected for the two different PDU Sessions. However, each SMF may also have the capability to control both a local and a central UPF within a PDU Session.

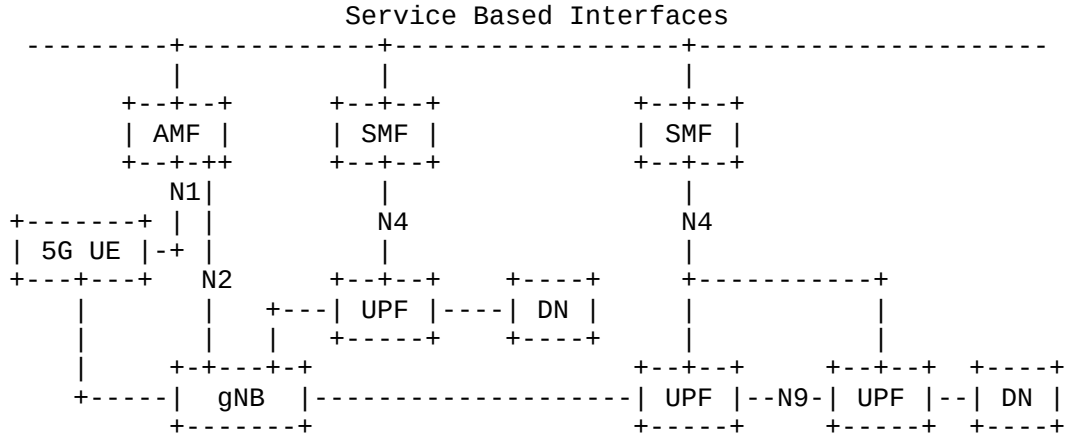


Figure 5: Non-roaming architecture for multiple PDU Sessions

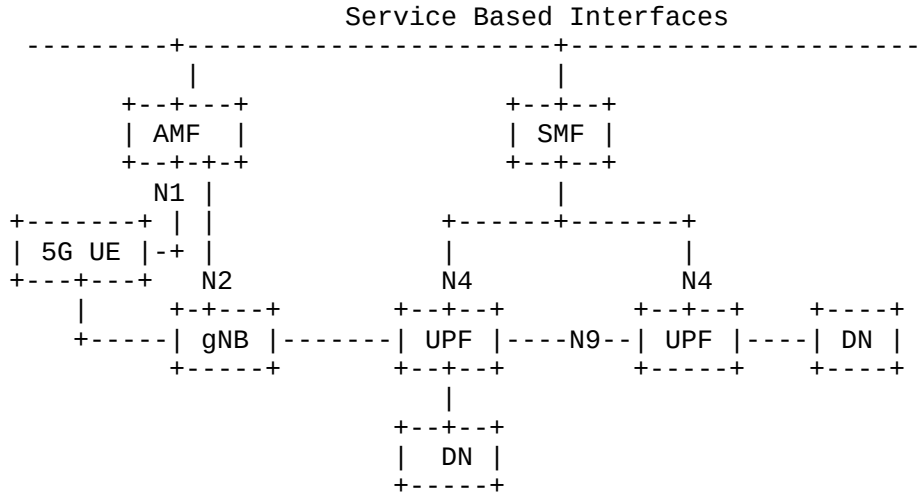


Figure 6: Non-roaming 5G System architecture for concurrent access to two (e.g. local and central) data networks (single PDU Session option)

Figure 6 depicts the non-roaming architecture in case concurrent access to two (e.g. local and central) data networks is provided within a single PDU Session.

The User plane function (UPF) is the function relevant to this evaluation and the N9 interface between two UPFs [23501].

The User Plane Function (UPF) handles the user plane path of PDU sessions. The UPF transmits the PDUs of the PDU session in a single tunnel between 5GC and (R)AN. The UPF includes the following functionality. Some or all of the UPF functionalities may be supported in a single instance of a UPF. Not all of the UPF functionalities are required to be supported in an instance of user plane function of a Network Slice.

- o Anchor point for Intra-/Inter-RAT mobility (when applicable)
- o Sending and forwarding of one or more "end marker" to the source NG-RAN node
- o External PDU Session point of interconnect to Data Network.
- o PDU session type: IPv4, IPv6, Ethernet, Unstructured (type of PDU totally transparent to the 5GS)
- o Activation and release of the UP connection of an PDU session,

upon UE transition between the CM-IDLE and CM-CONNECTED states (i.e. activation and release of N3 tunnelling towards the access network)

- o Data forwarding between the SMF and the UE or DN, e.g. IP address allocation or DN authorization during the establishment of a PDU session

- o Packet routing & forwarding (e.g. support of Uplink classifier to route traffic flows to an instance of a data network, support of Branching point to support IPv6 multi-homed PDU session)

- o Branching Point to support routing of traffic flows of an IPv6 multi-homed PDU session to a data network, based on the source Prefix of the PDU

- o User Plane part of policy rule enforcement, e.g. Gating, Redirection, Traffic steering)

- o Uplink Classifier enforcement to support routing traffic flows to a data network, e.g. based on the destination IP address/Prefix of the UL PDU

- o Lawful intercept (UP collection)

- o Traffic usage reporting e.g. allowing SMF support for charging, and/or allowing the SMF to initiate a CN initiated deactivation of UP connection of an existing PDU session when the UPF detects that the PDU Session has no user plane data activity for a specified Inactivity period provided by the SMF

- o QoS handling for user plane including:

  - packet filtering, gating, UL/DL rate enforcement, UL/DL Session-AMBR enforcement (with the Session-AMBR computed by the UPF over the Averaging window provisioned over N4, see subclause 5.7.3 of 3GPP TS 23.501), UL/DL Guaranteed Flow Bit Rate (GFBR) enforcement, UL/DL Maximum Flow Bit Rate (MFBR) enforcement, etc

  - marking packets with the QoS Flow ID (QFI) in an encapsulation header on N3 (the QoS flow is the finest granularity of QoS differentiation in the PDU session)

  - enabling/disabling reflective QoS activation via the User Plane, i.e. marking DL packets with the Reflective QoS Indication (RQI) in the encapsulation header on N3, for DL packets matching a QoS Rule that contains an indication to

activate reflective QoS

- o Uplink Traffic verification (SDF to QoS flow mapping, i.e. checking that QFIs in the UL PDUs are aligned with the QoS Rules provided to the UE or implicitly derived by the UE e.g. when using reflective QoS)

- o Transport level packet marking in the uplink and downlink, e.g. based on 5QI and ARP of the associated QoS flow

- o Packet Filter Set is used in the QoS rules or SDF template to identify a QoS flow. The Packet Filter Set may contain packet filters for the DL direction, the UL direction or packet filters that are applicable to both directions.

- o Downlink packet buffering and downlink data notification triggering:

  This includes the support and handling of the ARP priority of QoS Flows over the N4 interface, to support priority mechanism:

  - "For a UE that is not configured for priority treatment, upon receiving the "N7 PDU-CAN Session Modification" message from the PCF with an ARP priority level that is entitled for priority use, the SMF sends an "N4 Session Modification Request" to update the ARP for the Signalling QoS Flows, and sends an "N11 SM Request with PDU Session Modification Command" message to the AMF, as specified in clause 4.3.3.2 of TS 23.502.

  - "If an IP packet arrives at the UPF for a UE that is CM-IDLE over a QoS Flow which has an ARP priority level value that is entitled for priority use, delivery of priority indication during the Paging procedure is provided by inclusion of the ARP in the N4 interface "Downlink Data Notification" message, as specified in clause 4.2.3.4 of TS 23.502."

- o ARP proxying as specified in IETF RFC 1027 [53] and / or IPv6 Neighbour Solicitation Proxying as specified in IETF RFC 4861 [54] functionality for the Ethernet PDUs. The UPF responds to the ARP and / or the IPv6 Neighbour Solicitation Request by providing the MAC address corresponding to the IP address sent in the request.

- o Packet inspection (e.g. Application detection based on service data flow template and the optional PFDs received from the SMF in addition)

- Traffic detection capabilities
  For IP PDU session type, the UPF traffic detection capabilities may detect traffic using traffic pattern based on at least any combination of:
  PDU session
  QFI
  IP Packet Filter Set, comprising:
    - Source/destination IP address or IPv6 network prefix
    - Source / destination port number
    - Protocol ID of the protocol above IP/Next header type
    - Type of Service (TOS) (IPv4) / Traffic class (IPv6) and Mask
    - Flow Label (IPv6)
    - Security parameter index

  Application Identifier: The Application ID is an index to a set of application detection rules configured in UPF

  In the IP Packet Filter Set:
    - a value left unspecified in a filter matches any value of the corresponding information in a packet
    - an IP address or Prefix may be combined with a prefix mask
    - port numbers may be specified as port ranges

  For Ethernet PDU session type, the SMF may control UPF traffic detection capabilities based on at least any combination of:
  PDU session
  QFI
  Ethernet Packet Filter Set, comprising:
    - Source/destination MAC address
    - EtherType as defined in IEEE 802.3
    - Customer-VLAN tag (C-TAG) and/or Service-VLAN tag (S-TAG) VID fields as defined in IEEE 802.1Q
    - Customer-VLAN tag (C-TAG) and/or Service-VLAN tag (S-TAG) PCP/DEI fields as defined in IEEE 802.1Q
    - IP Packet Filter Set, in case Ethertype indicates IPv4/IPv6 payload
- Network slicing Requirements for different MM mechanisms on different slice.

  The selection mechanism for SMF to select UPF based on the selected network slice instance, DNN and other information e.g. UE subscription and local operator policies.

The following information is sent in an encapsulation header over the N3 interface. N9 needs to support that.

QFI (QoS Flow Identifier), see subclause 5.7.1 of 3GPP TS 23.501:

> "A QoS Flow ID (QFI) is used to identify a QoS flow in the 5G system. User Plane traffic with the same QFI within a PDU session receives the same traffic forwarding treatment (e.g. scheduling, admission threshold). The QFI is carried in an encapsulation header on N3 (and N9) i.e. without any changes to the e2e packet header. It can be applied to PDUs with different types of payload, i.e. IP packets, non-IP PDUs and Ethernet frames. The QFI shall be unique within a PDU session." The QFI is sent for both downlink and uplink user plane traffic.

RQI (Reflective QoS Identifier), see subclause 5.7.5.4.2 of 3GPP TS 23.501:

> "When the 5GC determines to activate reflective QoS via U-plane, the SMF shall include a QoS rule including an indication to the UPF via N4 interface to activate User Plane with user plane reflective. When the UPF receives a DL packet matching the QoS rule that contains an indication to activate reflective QoS, the UPF shall include the RQI in the encapsulation header on N3 reference point. The UE creates a UE derived QoS rule when the UE receives a DL packet with a RQI."
>
> The RQI is sent for downlink user plane traffic only.

Support of RAN initiated QoS Flow mobility, when using Dual connectivity, also requires the QFI to be sent within End Marker packets. See subclause 5.11.1 of 3GPP TS 23.501 and subclause 4.14.1 of 3GPP TS 23.502 respectively:

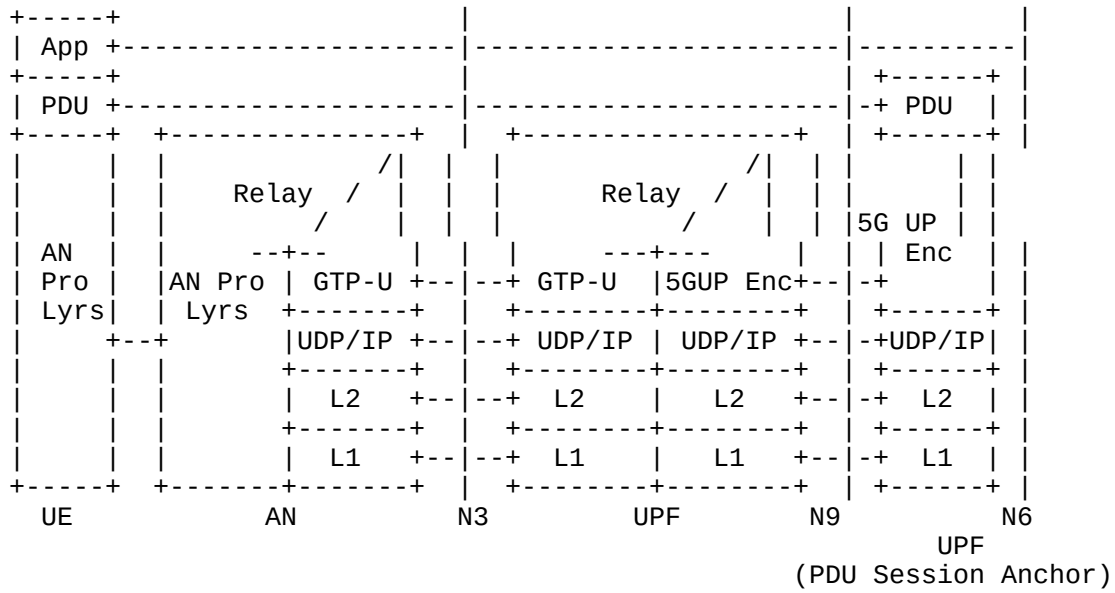
> " For some other PDU sessions of an UE: Direct the DL User Plane traffic of some QoS flows of the PDU session to the Secondary (respectively Master) RAN Node while the remaining QoS flows of the PDU session are directed to the Master (respectively Secondary) RAN Node. In this case there are, irrespective of the number of QoS Flows, two N3 tunnel terminations at the RAN for such PDU session."
>
> " In order to assist the reordering function in the Master RAN node and/or Secondary RAN node, for the QFIs that are switched between Master RAN node and Secondary RAN node, the UPF sends one or more "end marker" packets along with QFI on the old tunnel immediately after switching the tunnel for the QFI. The UPF starts sending downlink packets to the Target RAN."

GTPv1-U as defined in 3GPP TS 29.281 is used over the N3 and N9 interfaces in Release 15. Release 15 is still work-in-progress and RAN3 will specify the contents of the 5GS Container. It is to be decided whether CT4 needs to specify new GTP-U extension header(s) in

3GPP TS 29.281 for the 5GS Container.

A GTP-U tunnel is used per PDU session to encapsulate T-PDUs and GTP-U signaling messages (e.g. End Marker, Echo Request, Error Indication) between GTP-U peers.

A 5GS Container is defined as a new single GTP-U Extension Header over the N3 and N9 interfaces and the elements are added to this container as they appear with the forthcoming features and releases. This approach would allow to design the 5GS information elements independently from the tunneling protocol used within the 5GS, i.e. it would achieve the separation of the Transport Network Layer (TNL) and Radio Network Layer (RNL) as required in 3GPP TR 38.801 subclause 7.3.2. This would allow to not impact the RNL if in a future release a new transport network layer (TNL) other than GTP-U/UDP/IP (e.g. GRE/IP) was decided to be supported. The protocol stack for the User Plane transport for a PDU session is depicted below in Figure 7.



Legend:

- PDU layer: This layer corresponds to the PDU carried between the UE and the DN over the PDU session. When the PDU session Type is IPV6, it corresponds to IPv6 packets; When the PDU session Type is Ethernet, it corresponds to Ethernet frames; etc.

- GPRS Tunnelling Protocol for the user plane (GTP-U): This protocol supports multiplexing traffic of different PDU sessions (possibly corresponding to different PDU session Types) by

tunnelling user data over N3 (i.e. between the AN node and the UPF) in the backbone network. GTP shall encapsulate all end user PDUs. It provides encapsulation on a per PDU session level. This layer carries also the marking associated with a QoS Flow.

- 5G Encapsulation: This layer supports multiplexing traffic of different PDU sessions (possibly corresponding to different PDU session Types) over N9 (i.e. between different UPF of the 5GC). It provides encapsulation on a per PDU session level. This layer carries also the marking associated with a QoS Flow.

Figure 7: User Plane Protocol Stack

4 Reference Scenario(s) for Evaluation

Different proposals will be described for the following scenarios:

1. Non-Roaming Scenarios
   a. UE- Internet Connectivity (mobility cases)
   b. UE-UE IP Packet Flow (mobility cases)
   c. UE - 2 DNs with multiple PDU sessions
   d. UE - 2 DNs single PDU session

2. Roaming Scenarios
   a. Local Break out
   b. Home routed

Flows will be provided for mobility cases (UE mobility, UPF mobility) and session continuity cases (SSC Mode 1/2/3).

1. UE mobility SSC Mode 1
   a. Single UPF
   b. Multiple UPF
2. UE Mobility SSC Mode 2
   a. Single UPF
   b. Multiple UPF
3. UE Mobility SSC Mode 3
   a. Single UPF
   b. Multiple UPF

Each proposal will also describe how network slicing will be supported for the following configurations:

- Support for independent slices using GTP and/or other protocol will be covered. Mobility Management will be within each slice.

- Support for one UE connected to multiple slices using different mobility protocols will be described.

The criteria for evaluation will be the ability to support the above scenarios and identifying the impacts to N2, N3, N4, gNB, AMF and SMF.

5 SRv6 Based Solution

5.1 Overview

Segment Routing (SR), defined in [I-D.ietf-spring-segment-routing] generalises the source routing paradigm with an ordered list of global and/or nodal instructions (segments) prepended in an SR header in order to either steer traffic flows through the network while confining flow states to the ingress nodes in the SR domains and/or to indicate functions that are performed at specific network locations.

The IPV6 realisation of SR (SRV6) defines a SR Header (SRH), see [I-D.ietf-6man-segment-routing-header]. SRV6 encodes segments as IPV6 addresses in the Segment List (SL) of its header. The packet destination address in SRV6 specifies the active segment while an index field in the SRH points to the next active segment in the list. The index field in SRH is decremented as SRV6 progressively forces packet flows through different segments over the IPV6 data plane.

The versatility and adaptability of SR combined with IPV6’s ample and flexible address space positions SRV6 as a viable data path technology for the next generation of mobile user plane, in particular the 3GPP N9 (UPF to UPF).

This section starts by summarising the use of SRV6 as a drop-in alternative for GTP-U over the N9 interface connecting different User Plane Functions (UPF). It then shows how SRV6 as a GTP-U replacement can then provide additional features such as TE, sparing, rate limiting, and service chaining that are not natively available by GTP.

The discussion then focuses on advanced routing with the Identifier/Locator paradigm and shows how SRV6 can be used to realise this model in the mobile back-haul in either an anchored or anchorless mode of operation.

SRV6 appears well placed as a mechanism to replace GTP-U with initially no control plane changes, but to then offer a progressive path towards many innovations in routing.

5.2 SRV6 as Drop-In Alternative for GTP-U

Existing mobile back-haul employs GTP tunnels to carry user traffic

flows in the network. These tunnels are unidirectional, are established via the control plane for a particular QoS level, and run on links between access and the different anchor nodes all the way to DN gateways. 3GPP uses the term UPF to refer to the variety of functions performing different tasks on user traffic along the data path in 5G networks and suggests the use of GTP tunnels to carry user traffic between these UPFs (N9 interface).

The Tunnel Id (TEID) field in the GTP tunnel plays a crucial role in stitching the data path between the above mentioned network nodes for a particular user flow. In other words, TEIDs are used to coordinate traffic hand off between different UPFs.

In its most basic form, SRV6 can be used as a simple drop-in alternative for GTP tunnels. The control plane in this approach remains the same, and still attempts to establish GTP-U tunnels and communicate TEIDs between the tunnel end points. However, at the user plane, SRV6 capable nodes use SIDs to direct user traffic between the UPFs.

The simplest option is to encapsulate the entire GTP frame as a payload within SRV6. However, this scheme still carries the GTP header as the payload and as such doesn’t offer significant advantage.

A much more promising option however is to use SIDs to carry tunnel related information. Here, TEIDs and other relevant data can be encoded into SRV6 SIDs which can be mapped back to TEID’s at the intermediate UPFs thus requiring no changes except at the encapsulation and de-encapsulation points in the UPF chains.

[I-D.ietf-dmm-srv6-mobile-uplane] discusses the details of leveraging the existing control plane for distributing GTP tunnel information between the end nodes and employing SRV6 in data plane for UPF connectivity. The document defines a SID structure for conveying TEID, DA, and SA of GTP tunnels, shows how hybrid IPV4/IPV6 networks are supported by this model and in doing so, it paves a migration path toward a full SRV6 data plane.

Another alternative that can provide for a smooth migration toward SRV6 data plane between UPFs is via the use of "Tag", and optional TLV fields in SRH. Similar to the previously described method, this approach takes advantage of the existing control plane to deliver GTP tunnel information to the UPF endpoints. "Tag" and optional TLV fields in SRH are then used to encode tunnel information in the SRV6 data plane where the UPFs can determine the TEID etc. by inverting the mapping.

In yet another option, GTP tunnel information can be encoded as a separate SID either within the same SRH after the SID that identifies the UPF itself (SRH-UPF)or inside a separate SRH (SRH-G). In this option, SID representing the GTP tunnel information acts as both start and end point of a segment within the UPF. This option resembles the MPLS label stacking mechanism which is widely used in different VPN scenarios.

It must be noted that in any of the above mentioned approaches, the ingress UPF in SRV6 domain can insert a SRH containing the list of SIDs that corresponds to all UPFs along the path. Alternatively, UPFs can stack a new SRH on top of the one inserted by the previous one as packets traverse network paths between different pairs of UPFs in the network.

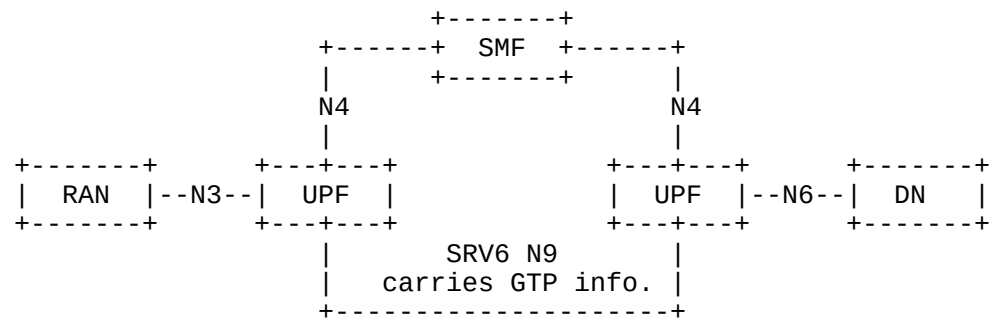


Figure 8: SRV6 as Drop-In replacement for GTP-U in 5G

5.3 SRV6 as Drop-In GTP Replacement with TE

The previous section discussed using SRV6 as a drop-in replacement for GTP tunnels in existing mobile networks. No new capabilities were introduced by this simple 1 to 1 replacement. We now explore additional possible features once SRV6 has been introduced.

Traffic engineering is an integral feature of SR. The SRV6 variant of SR of course supports both strict and loose models of source routing. Here, the SID list in SRH can represent a loose or strict path to UPFs. Therefore, traffic engineering can easily be supported regardless of any of the aforementioned approaches.

For loose paths to UPFs, a set of one or more SIDs in SRH’s SID list identifies on or more, but not all the intermediate nodes to a particular UPF. Packets then follow the IGP shortest path through the network to each specified intermediate node till they reach the target UPF.

In the case of strict path to UPFs, SRH contains a set of SIDs representing all the intermediate nodes and links that the packet must visit on its route to a particular UPF. The last SID in the set represents the target UPF itself or the last link to this UPF. Here, SRV6 packet processing at each node invokes the function(s) that is associated with SID[SL], the packet then receives the required treatment and gets forwarded over the SRH's specified path toward the target UPF.

It must be noted that the SRH could contain multiple sets of SIDs each representing a TE path between a pair of UPFs. Alternatively, the SRH can contain a fully resolved end to end TE path that covers every intermediate node and UPF along the data plane.

SR considers segments to be instructions. Therefore each SID can represent a function that enforces a specific nodal or global packet treatment. Attributes such as jitter and delay requirement, rate limiting factors, etc. can be easily encoded in to SIDs in order to apply the desired treatment as packets traverse the network from UPF to UPF. [I-D.ietf-dmm-srv6-mobile-uplane] suggests a SID encoding mechanism for rate limiting purposes.

Please refer to the followings for further details about SR and SRV6 traffic engineering capabilities, network programming concept, and a list of some of the main SR functions.

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

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

[I-D.filsfils-spring-srv6-network-programming].

[draft-gundavelli-dmm-mfa-00.txt]

5.4 UPF Chaining with SRV6

Service or function chaining is another intrinsic feature of SR and its SRV6 derivative. Using this capability, operators can direct user traffic through a set of UPFs where each UPF performs a specific task or executes certain functions on the traffic.

UPF chaining is achieved through the use of SIDs in SRV6 in the manner identical to what was described in the previous section regarding SRV6 support for traffic engineering.

Generally speaking, the SRH is populated with a set of SIDs with each SID identifying a specific UPF in the network. Starting from the ingress SRV6 node, packets are then forwarded through the network in

either loose or explicit mode toward each UPF.

Please refer to [I-D.xuclad-spring-sr-service-chaining] for further detail.

5.5 SRV6 and Entropy

Ability to provide a good level of entropy is an important aspect of data plane protocols. The TEID field in GTP tunnels if included in network node's hashing algorithms can result in good load balancing. Therefore, any new data plane proposal should be able to deal with entropy in an efficient manner.

SRV6 SIDs can easily accommodate entropy at either hop by hop or global level via reserving a set of bits in the SID construct itself; and hence, eliminate the need for a special entropy Segment ID in SRH. Here, the hashing algorithm at different nodes distribute traffic flows based on the SID which has been copied to IPV6 DA field.

Alternatively, entropy related information can be encoded as optional TLV field in SRV6's SRH.

5.6 SRV6 and 5G Slicing

Slicing is one of the main features in 5G [3GPP 23501]. Several Slices with different requirements can coexist on top of the common network infrastructure. Diverse flows belonging to different 5G slices can be completely disjoint or can share different parts of the network infrastructure. SRV6's native features such as TE, Chaining, one-plus-one protection, etc. either in stand-alone or in conjunction with other alternatives for mobility support such as ID-LOC model lend themselves well to 5G slicing paradigm.

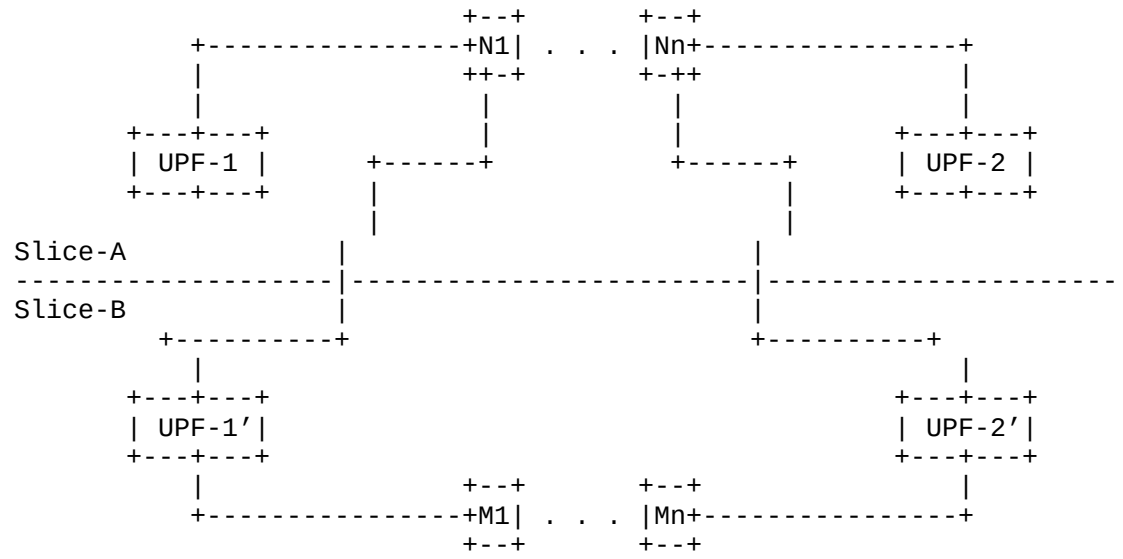


Figure 9: SRV6 TE, Service Chaining, Sparing, and Protection for 5G Slices

5.7 SRV6 and Lawful Interception in 5G

To be filled.

5.8 SRV6 and Alternative Approaches to Advanced Mobility Support

SRV6 flexibility enables it to support different methods of providing mobility in the network. ID-LOC for mobility support is one such option.

5.8.1 SRV6 and Locator/ID Separation Paradigm for N9 Interface

The previous sections discussed how SRV6 could be employed as a replacement for GTP tunnels while leaving the existing control plane intact. This section describes the use of SRV6 as a vehicle to implement Locator/ID Separation model for UPF data plane connectivity.

5.8.2 Brief Overview of Locator-ID Separation

Traditional routing architecture uses IP addresses as both device identity and its location in the network. Locator-ID Separation model establishes a paradigm in which a device identity and its network location are split into two separate namespaces: End-point Identifiers (EID), and Route Locators (RLOC) that are correlated via

a control plane, or a dynamic (centralised or distributed) mapping system.

RLOCs are tied to network topology. They represent network devices that are reachable via traditional routing. EIDs, on the other hand, represent mobile or stationary devices, functions, etc. that are reachable via different RLOCs based on the network location where they get instantiated, activated or moved.

Using this model, as long as EID-RLOC relationship remains up to date, EIDs can easily move between the RLOCs. That is the EID namespace can freely move without any impact to the routing paths and connectivity between the Route Locators.

This type of multi encapsulation and routing has been employed in fixed networks (IP, VPN, MPLS, etc.). The use of this paradigm in mobile data plane, therefore, offers an approach that takes advantage of a mature and proven technology to implement the N9 interface for UPF connectivity.

5.8.3 Locator-ID Separation via SRV6 for UPF connectivity

SRV6 can easily implement ID-LOC Separation model for UPF connectivity. The SIDs are once again the main vehicle here. In this model, UPFs are considered to be the IDs while the nodes where the UPFs attach to take on the role of the Locators. Multiple UPFs are allowed to attach to the same Locator. It is also possible for a UPF to connect to multiple Locators. There are several implementation options. The followings highlights a few possibilities.

5.8.3.1 Overlay model with SRV6 Locators

In this approach, UPFs connect to SRV6 capable Locators. UPFs use IPV4/IPV6 transport either in conjunction with GTP or without any GTP tunnel and send the packets to their associated Locator at the near end (Ingress SRV6 Locator).

In either case, the ingress SRV6 Locator uses the DA field in arriving packets to identify the far end Locator (Egress SRV6 Locator) where the target UPF is attached and obtains its associated SID.

For GTP encapsulated traffic from UPFs, the ingress SRV6 Locator must also deliver GTP information to the far end Locator. Please see section 5.2. for more information on different methods of conveying GTP information in SRV6 domains.

The ingress SRV6 Locator then constructs the SRH and sends the

traffic through the SRV6 network toward the egress SRV6 Locator. Egress Locator marks the end of the segment and ships the traffic to the target UPF.

It must be noted that use of GTP at UPFs allows us to leave the 3GPP control plane intact and hence provides a smooth migration path toward SRV6 with ID-Locator model. For inter UPF traffic that doesn't use GTP, the control plane requires some modifications in order to be able to convey endpoint information to interested parties.

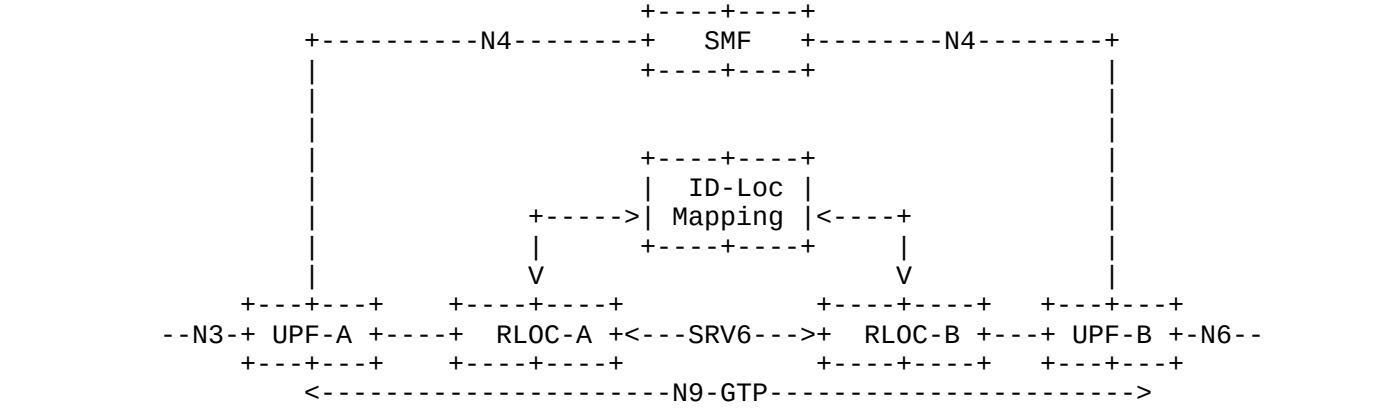


Figure 10: Overlay Model with SRV6 Locator in 5G

5.8.3.2 SRV6 Capable UPFs and RLOCs

In this model, the head end UPF (Ingress UPF) is the ingress node and the entity that constructs the SRH in the SRV6 domain. Here, both UPFs (IDs) and Locators are represented by SIDs in the SRH. The SID list establishes either a partial or the full path to a target or a set of UPFs that traffic is required to traverse.

The 3GPP control plane is responsible for distributing UPF's endpoint information. But it requires some modifications to be able to convey endpoint information to interested parties.

In its simplest form, the SMF using policy information prepares a set of one or more UPFs along the traffic path and distributes this set in the form a SID list to the ingress UPF. This SID list of UPFs is then gets augmented with a set of SIDs identifying the Locators representing the current point of attachment for each UPF along the data path.

Alternatively, the SMF can provide a fully resolved SID list by communicating with a centralised or distributed ID-LOC mapping system containing all the relevant data regarding the UPF-Locator

relationship.

In yet another approach, the SMF can provide a partial SID list representing the segment between each pair of UPFs to individual UPFs along the path.

Regardless of the approach, any changes to UPF's point of attachment must be reflected in the mapping system and communicated to the SMF for distribution to the appropriate set of UPFs. Keeping the mapping system current is essential to proper operation. As long as the mapping database is up-to-date, UPFs can be easily moved in the network. Design of ID-Locator mapping system is beyond the scope of this document. However, experiment with distributed mapping systems offered by today's public clouds has shown very promising results which can be further improved and tailored to mobile network requirements.

The following figure shows the use of SRV6 UPFs and RLOCs in 5G.

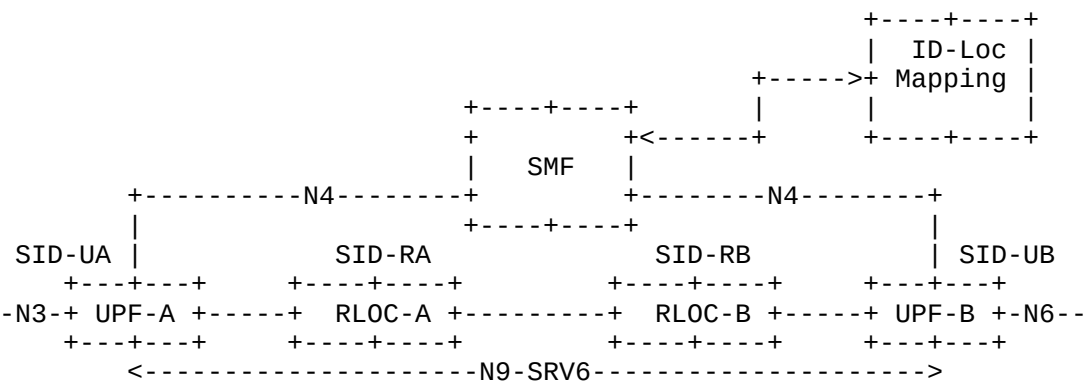


Figure 11: SRV6 Capable UPFs and Locators in 5G

5.8.4 Advanced Features in ID-Locator Architecture

SRV6's native features such as Traffic Engineering, QoS support, UPF Chaining, etc. can be easily added to ID-Locator support. As it was noted earlier, these features are not readily available by GTP.

5.9 Areas of Concern

Support for IPV6 is a precondition for SRV6. Although SRV6 can support hybrid IPV4/IPV6 mobile data plane through an interworking node, support of UPFs with IPV4 address is rather complex.

Due to IPV6 128-bit address space, large SRH size can have a negative

impact on MTU. Large SRH size can also exert undesirable header tax especially in the case of small payload size. Furthermore, compound SID processing at each node might affect line rate.

ID-LOC architecture relies on high performance mapping systems. Distributed mapping systems using some form Distributed Hash Table(DHT) exhibit very promising results. But further investigation is required to ensure mobility requirements in mobile data plane.

6 LISP based Solution

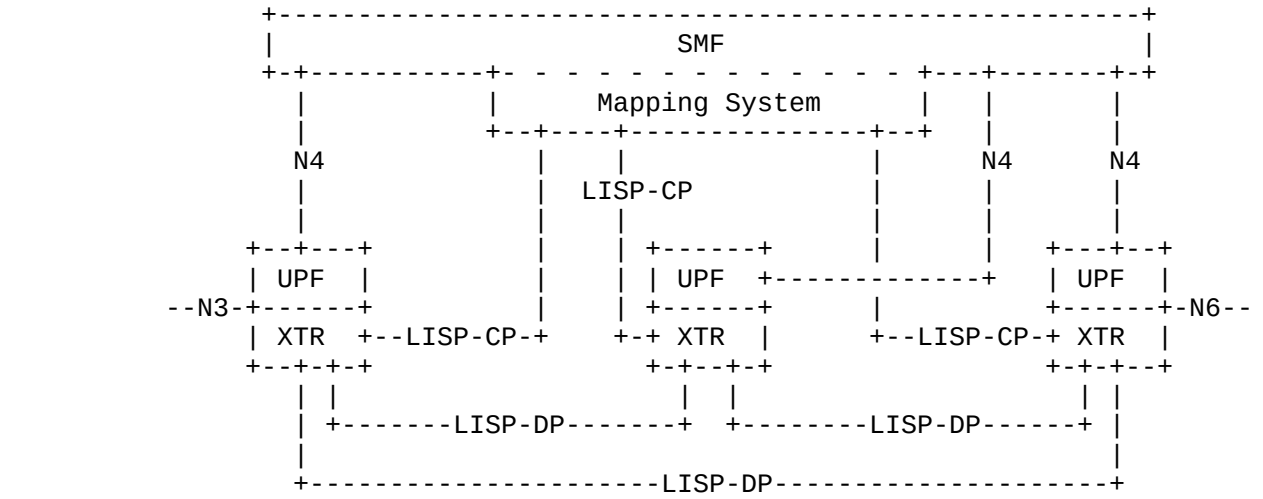


Figure 12: LISP in the 5G architecture

6.1 Overview

The Locator/Identifier Separation Protocol (LISP), which provides a set of functions for routers to exchange information used to map from Endpoint Identifiers (EIDs) that are not globally routable to routable Routing Locators (RLOCs). It also defines a mechanism for these LISP routers to encapsulate IP packets addressed with EIDs for transmission across a network infrastructure that uses RLOCs for routing and forwarding.

An introduction to LISP can be found in [I-D.ietf-lisp-introduction].

A complete RFC-set of specifications can be found in [RFC6830], [RFC6831], [RFC6832], [RFC6833], [RFC6836], [RFC7215], [RFC8061], [RFC.8111]. They describe support and mechanisms for all combinations of inner and outer IPv4 and IPv6 packet headers for unicast and multicast packet flows that also interwork with non-LISP sites as

well as two designs to realize a scalable mapping system.

A standards-track based set of drafts [I-D.ietf-lisp-rfc6830bis] [I-D.ietf-lisp-rfc6833bis] are products and work in progress of the LISP Working Group.

6.2 LISP Data-Plane

LISP uses dynamic tunnel encapsulation as its fundadmental mechanism for the data-plane. Fixed headers are used between the outer and inner IP headers which are 16 bytes in length. Details can be found in[RFC6830].

6.3 LISP Control-Plane

Many years of research dating back to 2007 have gone into LISP scalable mapping systems. They can be found at [LISP-WG] and [IRTF-RRG]. The two that show promise and have deployment experience are LISP-DDT [RFC8111] and LISP-ALT [RFC6836].

The control-plane API which LISP xTRs are the clients of is documented in [RFC6833]. Various mapping system and control-plane tools are available [RFC6835] [RFC8112] and are in operational use.

6.4 LISP Mobility Features

LISP supports multi-homed shortest-path session survivable mobility. An EID can remain fixed for a node that roams while its dynamic binding changes to the RLOCs it uses when it reconnect to the new network location.

When the roaming node supports LISP, its EIDs and RLOCs are local to the node. This form of mobility is call LISP Mobile-Node. Details can be found in [I-D.ietf-lisp-mn].

When the roaming node does not support LISP, but LISP runs in the network the node roams to, the EIDs and RLOCs are not co-located in the same device. In this case, EIDs are assigned to the roaming node and RLOCs are assigned to LISP xTRs. So when the roaming node attaches to the network, its EIDs are mapped to the RLOCs of the LISP xTRs in the network. This form of mobility is called LISP EID-Mobility. Details can be found in [I-D.ietf-lisp-eid-mobility].

For a 3GGP mobile network, the LISP EID-Mobility form of mobility is recommended and is specified in the use-case document [I-D.ietf-farinacci-lisp-mobile-network].

6.5 ILSR

ILSR is a specific recommendation for using LISP in the 3GPP 5G mobile network architecture. A detailed whitepaper can be found at [ILSR-WP]. The recommendation is to use the mechanisms in [I-D.ietf-farinacci-lisp-mobile-network].

6.6 LISP Control-Plane with ILA Data-Plane

In the LISP WG re-charter of 2016, consensus was reached to separate the data-plane and control-plane aspects of the protocol. The current LISP control-plane (LISP-CP) specification [I-D.ietf-lisp-rfc6833bis] is data-plane agnostic and can serve as control-plane for different data-plane protocols. In this section we describe how LISP-CP can serve to enable the operation of an ILA data-plane. A similar approach can be followed to use LISP-CP as control-plane for other data-plane protocols (e.g. VXLAN, SRv6, etc).

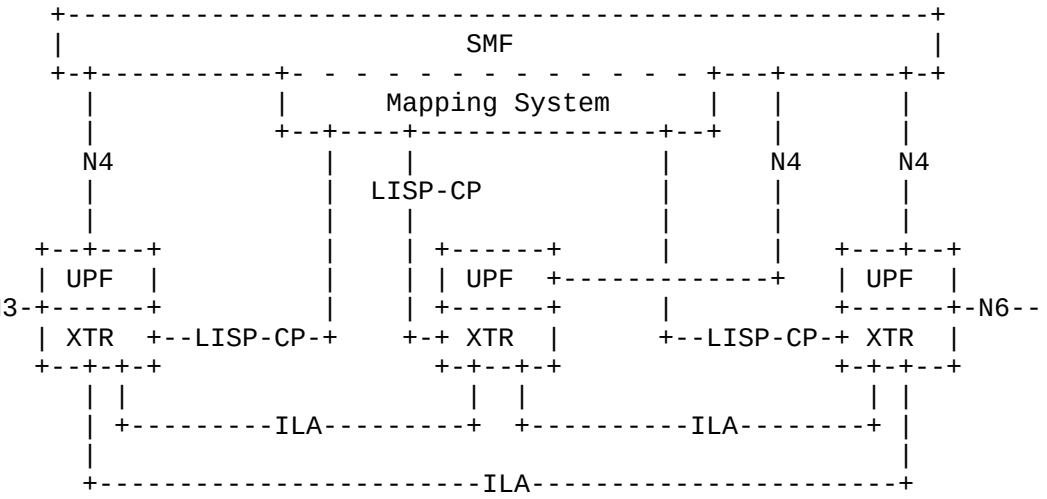


Figure 13: LISP-CP + ILA in the 5G architecture

Please refer to Section 8 for description of the ILA data-plane. The complete specification of how to use the LISP-CP in conjunction with an ILA data-plane can be found in [I-D.rodrigueznatal-ila-lisp]. Below are summarized the major points to take into account when running LISP-CP as control-plane for ILA.

- Leveraging on the flexible LISP-CP address encoding defined in [RFC8060], different ILA address types are defined in [I-D.rodrigueznatal-ila-lisp] to carry ILA metadata over the LISP-CP.

- XTRs can serve as both ILA-Ns (when their map-cache is incomplete) or ILA-Rs (when their map-cache is complete). XTRs serving as ILA-Rs subscribe to the Mapping System to populate their map-cache with all the mappings in the domain (or its shard) using [I-D.rodrigueznatal-lisp-pubsub].

- LISP-CP can run over TCP or UDP. The same signaling and logic applies independently of the transport. Additionally, when running over TCP, the optimizations specified in [I-D. kouvelas-lisp-map-server-reliable-transport] can be applied.

- The ILA control-plane operations "request/response" and "push" are implemented via the LISP mechanisms defined in [I-D.ietf-lisp-rfc6833bis] and [I-D.rodrigueznatal-lisp-pubsub] respectively. When the Mapping System is co-located with the XTRs serving as ILA-Rs, the ILA "redirect" operation is implemented via the mapping notifications described in [I-D.rodrigueznatal-lisp-pubsub].

- XTRs serving as ILA-Ns can use LISP-CP as described in [I-D.ietf-lisp-rfc6833bis] to register and keep updated in the Mapping System the information regarding their local mappings.

- When using ILA as data-plane, the mobility features and benefits discussed in Section 8 and in [I-D.ietf-lisp-eid-mobility] still apply.

- As discussed in [I-D.rodrigueznatal-ila-lisp], the LISP-CP can be used not only to resolve ID-Loc mappings but also to obtain the ILA Identifier when it is not possible to locally derivate it from the endpoint address. These two mapping operations can be combined into one to obtain the ILA Identifier and associated locators in a single round of signaling.

7 ILNP Based Solution

<Text to be Included>.

8 ILA based Solution

Identifier-Locator Addressing [ILA] is a protocol to implement transparent network overlays without encapsulation. It addresses the need for network overlays in virtualization and mobility that are efficient, lightweight, performant, scalable, secure, provide seamless mobility, leverage and encourage use of IPv6, provide strong privacy, are interoperable with existing infrastructure, applicable to a variety of use cases, and have simplified control and management.

8.1 Overview of ILA

ILA is a form of identifier/locator split where IPv6 addresses are transformed from application-visible, non-topological "identifier" addresses to topological "locator" addresses. Locator addresses allow packets to be forwarded to the network location where a logical or mobile node currently resides or is attached. Before delivery to the ultimate destination, locator addresses are reverse transformed back to the original application visible addresses. ILA does address "transformation" as opposed to "translation" since address modifications are always undone. ILA is conceptually similar to ILNP and 8+8, however ILA is contained in the network layer. It is not limited to end node deployment, does not require any changes to transport layer protocols, and does not use extension headers.

ILA includes both a data plane and control plane. The data plane defines the address structure and mechanisms for transforming application visible identifier addresses to locator addresses. The control plane’s primary focus is a mapping system that includes a database of identifier to locator mappings. This mapping database drives ILA transformations. Control plane protocols disseminate identifier to locator mappings amongst ILA nodes.

The use cases of ILA include mobile networks, datacenter virtualization, and network virtualization. A recent trend in the industry is to build converged networks containing all three of these to provide low latency and high availability. A single network overlay solution that works across multiple use cases is appealing.

Benefits of ILA include:

- Facilitates node mobility and virtualization
- Multiple use cases (mobile, datacenter, cloud)
- Super efficient and performant data plane
- Allows strong privacy in addressing [ADDRPRIV]
- Promotes anchorless mobility
- No typical tunneling issues (e.g. MTU) or management related to encapsulation
- Flexible control plane that splits data & control
- Modern "SDN" control protocols (e.g. RPC/TCP)

- o Scale number of nodes to billions for 5G, DC virtualization
- o Upstream Linux kernel data path [ILAKERNEL] and open source ctrl plane [ILACONTROL].

The ILA data plane protocol is described in [ILA], motivation and problems areas are described in [ILAMOTIVE], ILA in the mobile user-plane is described in detail in [ILAMOBILE].

8.2 ILA in the 5G architecture

ILA is a proposed alternative to GTP-U and encapsulation. It does not require anchors and simplifies both the data plane and control plane. ILA is a general network overlay protocol can be used to meet the requirements of use cases in a converged network. User Plane Functions (UPF) with ILA are lightweight and stateless such that they can be brought up quickly as needed.

Figures 13 and 14 depict two architectural options for the use of ILA in a 5G architecture. ILA is logically a network function and ILA interfaces to the 5G control plane via service based interfaces. In this architecture, ILA replaces GTP use over the N9 interface. Identifier address to locator address transformations in the downlink from the data network are done by an ILA-R. Transformations for intra domain traffic can be done by an ILA-N close to the gNB or by an ILA-R in the case of a cache miss. Locator address to identifier address transformation happen at ILA-Ns. ILA could be supported on a gNB. In this case, an ILA-N would be co- resident at a gNB and ILA is used over N3 interface in lieu GTP-U. Figures 14 and 15 depict two options of how ILA can be used in the 5G architecture. The control plane functions can be implemented as standalone network functions or can be implemented with other network functions. The control plane protocol can be implemented as enhancement to N4, as APIs or as independent protocol. Use of ILA in roaming scenarios is still TBD.

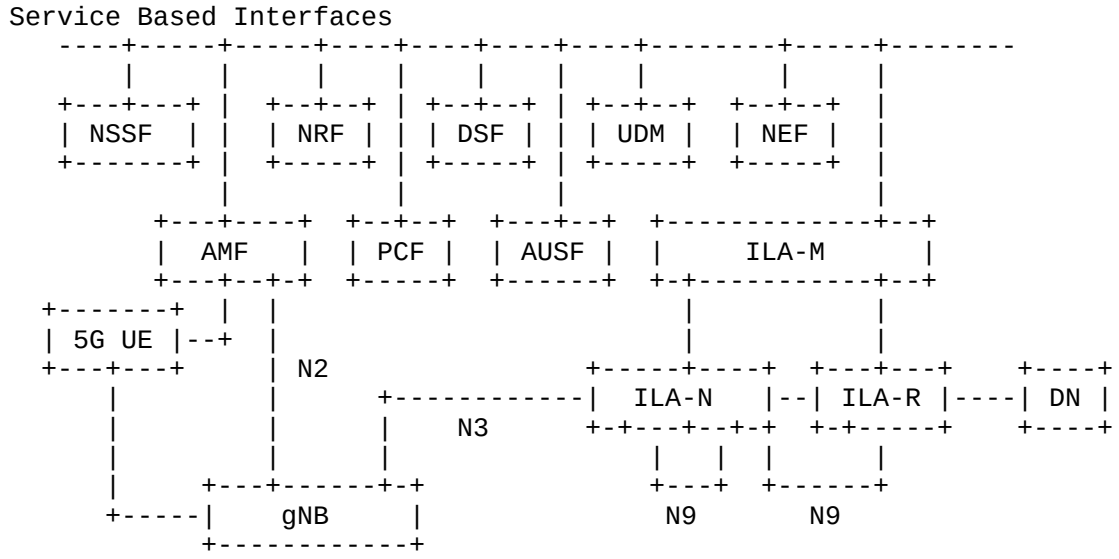


Figure 14: ILA in 5G architecture - Option 1

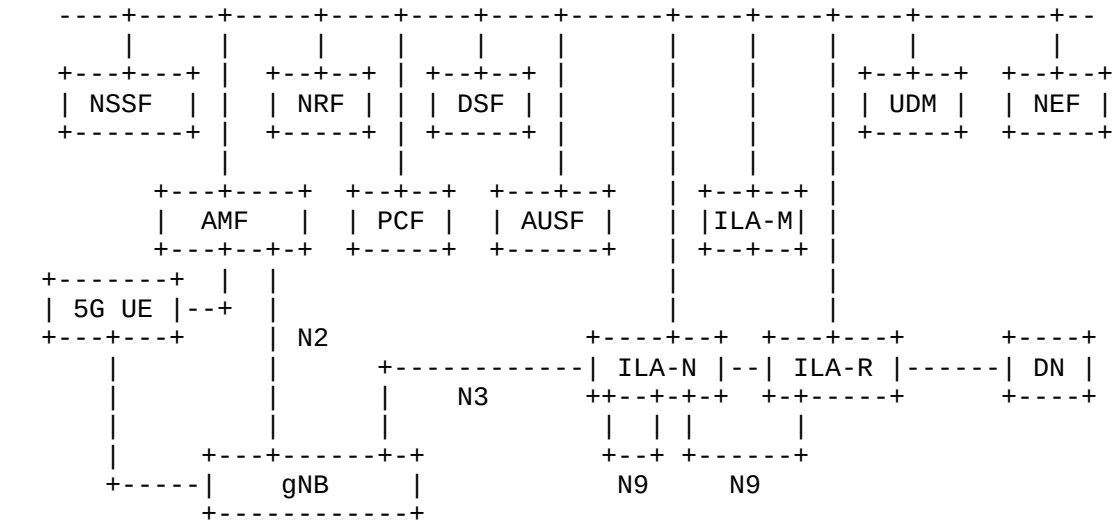


Figure 15: ILA in 5G architecture - Option 2

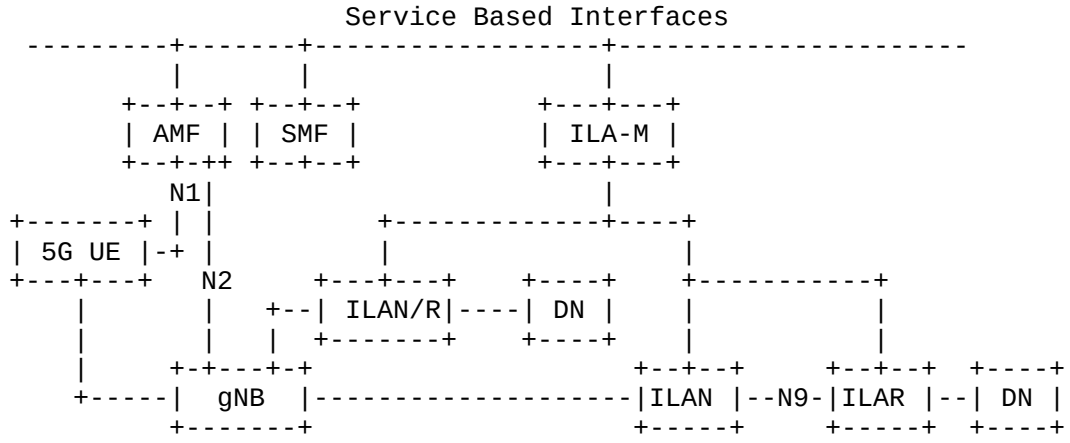


Figure 16: Non-roaming ILA-based architecture for multiple PDU Sessions

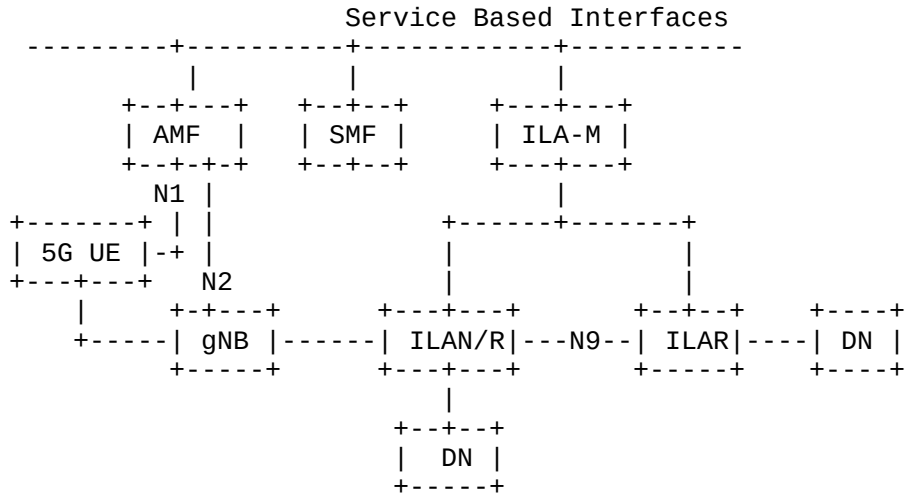


Figure 17: Non-roaming 5G ILA-based System architecture for concurrent access to two (e.g. local and central) data networks (single PDU Session option)

8.3 Protocol layering

Figure 3 illustrates the protocol layers of packets packets sent over various data plane interfaces in the downlink direction of data network to a mobile node. Note that this assumes the topology shown in Figure 2 where GTP-U is used over N3 and ILA is used on N9.

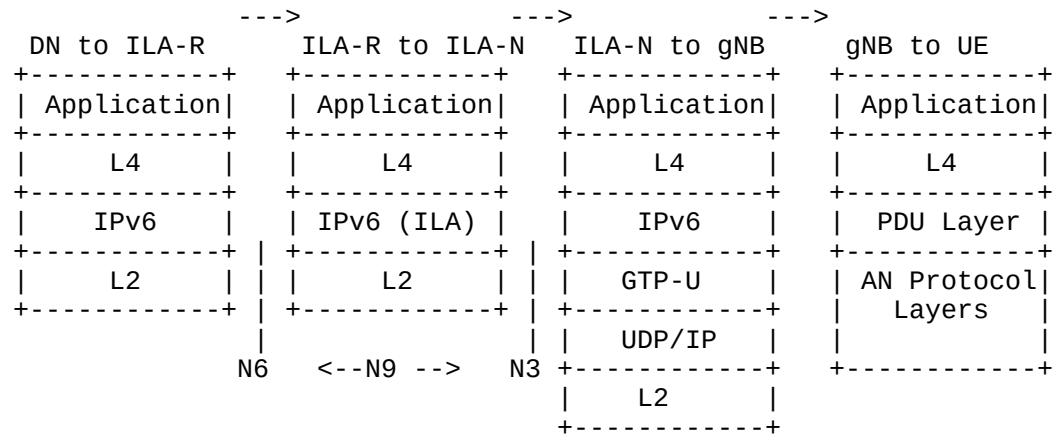


Figure 18: ILA and protocol layer in 5G

8.4 Control plane

ILA-M provides the interface between the 5G services architecture and the common ILA control plane.

8.4.1 ILA-M services interface

The control interface into ILA is via an ILA-M that interacts with 5G network services. ILA-M uses RESTful APIs to make requests to network services. An ILA-M receives notifications when devices enter the network, leave it, or move within the network. The ILA-M writes the ILA mapping entries accordingly.

ILA is a consumer of several 5G network services. The service operations of interest to ILA are:

- Nudm (Unified Data Management): Provides subscriber information.
- Nsmf (Service Managment Function): Provides information about PDU sessions.
- Namf (Core Access and Mobility Function): Provides notifications of mobility events.

8.4.2 ILA control plane

The ILA control plane is composed of mapping protocols that manage and disseminate information about the mapping database. There are two levels of mapping protocols: one used by ILA routers that require the full set of ILA mappings for a domain, and one used by ILA nodes that maintain a caches of mappings.

The ILA mapping system is effectively a key/value datastore that maps identifiers to locators. The protocol for sharing mapping information amongst ILA routers can thus be implemented by a distributed database [ILAMP]. ILA separates the control plane from the data plane, so alternative control plane protocols may be used with a common data plane [ILABGP],[ILALISP].

The ILA Mapping Protocol [ILAMP] is used between ILA forwarding nodes and ILA mapping routers. The purpose of the protocol is to populate and maintain the ILA mapping cache in forwarding nodes. ILAMP defines redirects, a request/response protocol, and a push mechanism to populate the mapping table. Unlike traditional routing protocols that run over UDP, this protocol is intended to be run over TCP and may be RPC oriented. TCP provides reliability, statefulness implied by established connections, ordering, and security in the form of TLS. Secure redirects are facilitated by the use of TCP. RPC facilities such REST, Thrift, or GRPC leverage widely deployed models that are popular in SDN.

8.5 IP addressing

ILA supports single address assignments as well as prefix assignments. ILA will also support strong privacy in addressing [ADDRPRIV].

8.5.1 Singleton address assignment

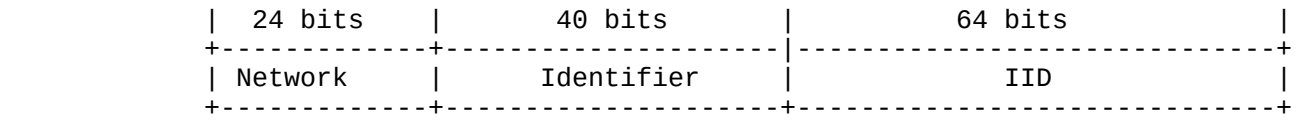
Singleton addresses can use a canonical 64/64 locator/identifier split. Singleton addresses can be assigned by DHCPv6.

8.5.2 Network prefix assignment

Prefix assignment can be done via SLAAC or DHCPv6-PD.

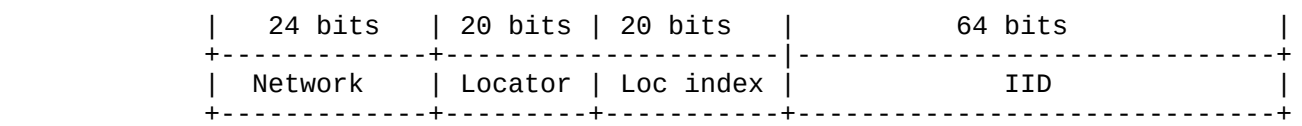
To support /64 prefix assignment with ILA, the ILA identifier can be encoded in the upper sixty-four bits of an address. A level of indirection is used so that ILA transforms the upper sixty four bits to contain both a locator and an index into a locator (ILA-N) specific table. The entry in the table provides the original sixty-four bit prefix so that locator to identifier address transformation can be done.

As an example of this scheme, suppose network has a /24 prefix. The identifier address format for /64 assignment might be:



The IID part is arbitrarily assigned by the device, so that is ignored by ILA. All routing, lookups, and transformations (excepting checksum neutral mapping) are based on the upper sixty-four bits.

For identifier to locator address transformation, a lookup is done on the upper sixty-four bits. That returns a value that contains a locator and a locator table index. The resulting packet format may be something like:



The packet is forwarded and routed to the ILA-N addressed by locator (/44 route in this case). At the ILA forwarding node, the locator index is used as a key to an ILA-N specific table that returns a 40 bit Identifier. This value is then written in the packet do ILA to identifier address transformation thereby restoring the original destination address.

The locator index is not globally unique, it is specific to each ILA-N. When a node attaches to an ILA-N, an index is chosen so that the table is populated at the ILA-N and the ILA mapping includes the locator and index. When a node detaches from on ILA, it’s entry in the table is removed and the index can be reused after a hold-down period to allow stale mappings to be purged.

8.5.3 Strong privacy addresses

Note that when a /64 is assigned to UEs, the assigned prefix may become a persistent identifier for a device. This is a potential privacy issue. [ADDPRIV] describes this problem and suggests some solutions that may be used with ILA.

8.6 Traffic engineering

ILA is primarily a mechanism for mobility and network virtualization. Transport mechanisms for traffic engineering such as MPLS, network slices, encapsulation, routing based on flow hash(flow label) can be applied independently of ILA. This separation allows any discussion related to transport to be left to operator deployment.

8.7 Locator Chaining with ILA

ILA transformations can be performed on a hop-by-hop bases. In this manner a packet can be source routed through a sequence of nodes. At each hop a determination is made as to the next hop the packet should visit. The locator for the target is then written into the destination. Eventually, the packet will be forwarded to an ILA forwarding node that will restore the original address before delivery to the final destination.

8.8 ILA and network slices

Figure 19 illustrates the use of network slices with ILA.

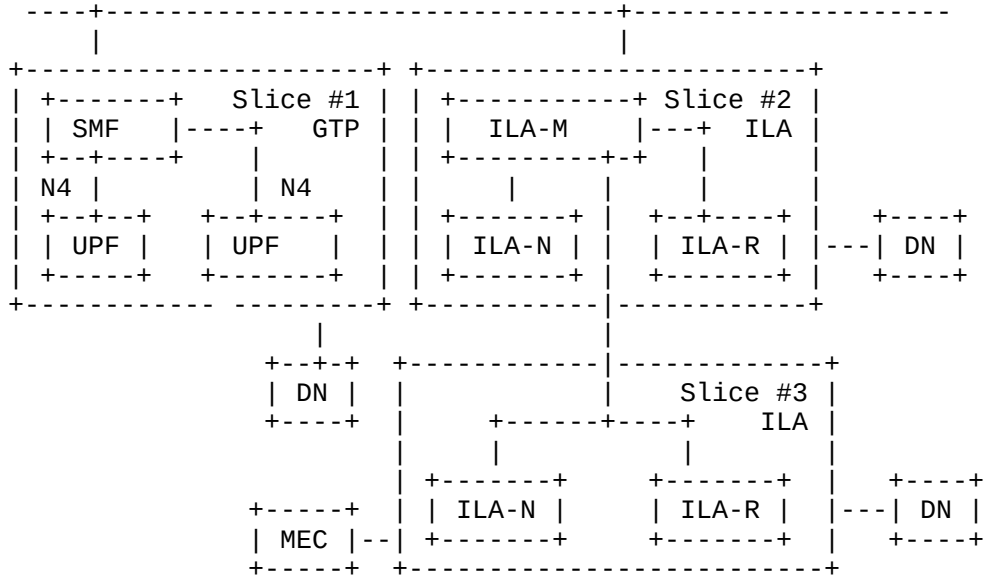


Figure 19: ILA and network slices in 5G

In this figure, slice #1 illustrates legacy use of UPFs without ILA in a slice. ILA can be deployed incrementally or in parts of the network. As demonstrated, the use of network slices can provide domain isolation for this.

Slice #2 supports ILA. Some number of ILA-Ns and ILA-Rs are deployed. ILA transformations are performed over the N9 interface. ILA-Rs would be deployed at the N6 interface to perform transformations on packets received from a data network. ILA-Ns will be deployed deeper in the network at one side of the N3 interface. ILA-Ns may be supplemented by ILA-Rs that are deployed in the network. ILA-M manages the ILA

nodes and mapping database within the slice.

Slice #3 shows another slice that supports ILA. In this scenario, the slice is for Mobile Edge Computing. The slice contains ILA-Rs and ILA-Ns, and as illustrated, it may also contain ILA_Hs that run directly on edge computing servers. Note in this example, one ILA-M, and hence one ILA domain, is shared between slice #2 and slice #3. Alternatively, the two slices could each have their own ILA-M and define separate ILA domains.

8.9 Security considerations

A mobile public infrastructure has many considerations in security as well as privacy. Fundamentally, a system must protect against misdirection for the purposes of hijacking traffic, spoofing, revealing user identities, exposing accurate geo-location, and Denial of Service attacks on the infrastructure.

The ILA mapping system contains personally identifiable information (PII) including user identities and geo-location. The information must be safeguarded. An ILA domain is confined to one administrative domain, only trusted parties entities in the domain participate in ILA. There is no concept of a global, public mapping system and UEs in public networks generally do not participate in ILA protocols since they are untrusted. ILA control protocols, include ILA redirects, use TCP. TLS or other protocols can be applied for strong security.

Privacy in addressing is a consideration. ILA endeavors to provide a mechanism of address assignment that prevents inference of user identity or location. This problem is described in [ADDRPRIV].

9 No Protocol Option

In this option, mobility is handled nomadically by the app.

10 Comparison of Protocols

This section will compare the different protocols with reference to how they will support the requirements for UPF and N9 interface; how the various scenarios identified in Sections 3 and 4 will be supported and impacts to other interfaces and functions of the architecture (e.g. N3, N4, SMF, AMF, etc).

11 Summary

This document summarized the various IETF protocol options for GTP replacement on N9 interface of 3GPP 5G architecture.

12 Formal Syntax The following syntax specification uses the augmented Backus-Naur Form (BNF) as described in RFC-2234 [RFC2234].

<Define your formal syntax here.>

13 Security Considerations

<Add any security considerations>

14 IANA Considerations

<Add any IANA considerations>

15 References

15.1 Normative References

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

15.2 Informative References

[ILA] Herbert, T., and Lapukhov, P., "Identifier Locator Addressing for IPv6" draft-herbert-intarea-ila-00

[ILAMP] Herbert, T., "Identifier Locator Addressing Mapping Protocol" draft-herbert-ila-ilamp-00

[29891] 5G System - Phase 1, CT WG4 Aspects, 3GPP TR 29891 v15.0.0, December 2017

[29244] Interface between the Control Plane and the User Plane Nodes; Stage 3, 3GPP TS 29.244 v15.0.0, December 2017

[29281] GPRS Tunneling Protocol User Plane (GTPv1-U), 3GPP TS 29.281 v15.1.0, December 2017

[23501] System Architecture for 5G System; Stage 2, 3GPP TS 23.501 v2.0.1, December 2017

[23502] Procedures for 5G System; Stage 2, 3GPP TS 23.502, v2.0.0, December 2017

[23503] Policy and Charging Control System for 5G Framework; Stage 2, 3GPP TS 23.503 v1.0.0, December 2017

[38300] NR and NG-RAN Overall Description: Stage 2, 3GPP TS 38.300 v2.0.0, December 2017

[38401] NG-RAN: Architecture Description, 3GPP TS 38.401 v1.0.0, December 2017

[CT4SID] Study on User Plane Protocol in 5GC, 3GPP CP-173037, December 2017

[RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, DOI 10.17487/RFC3513, April 2003, <https://www.rfc-editor.org/info/rfc3513>.

[RFC4941] Kolkman, O. and R. Gieben, "DNSSEC Operational Practices", RFC 4641, DOI 10.17487/RFC4641, September 2006, https://www.rfc-editor.org/info/rfc4641

[ILAGRPS] Herbert, T., "Identifier Groups in ILA", To be published

[BGPILA] Lapukhov, P., "Use of BGP for dissemination of ILA mapping information" draft-lapukhov-bgp-ila-afi-02

[3GPPTS] 3rd Generation Partnership Project (3GPP), "3GPP TS 23.502", http://www.3gpp.org/DynaReport/23-series.htm

Acknowledgments

The author would like to thank Farooq Bari, Devaki Chandramouli, Ravi Guntupalli, Sri Gundavelli, Peter Ashwood Smith, Satoru Matsushima, Michael Mayer, Vina Ermagan, Fabio Maino, Albert Cabellos, and Cameron Byrne for reviewing various iterations of the document and for providing content into various sections.

Authors' Addresses

K. Bogineni Verizon

Email: kalyani.bogineni@verizon.com

A, Akhavain Huawei Technologies Canada

Email: arashmid.akhavain@huawei.com

T. Herbert Quantonium Email: tom@quantonium.net

D. Farinacci lispers.net Email: farinacci@gmail.com

A. Rodriguez-Natal Cisco

Email: natal@cisco.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: March 22, 2018

D. Farinacci
lispers.net
P. Pillay-Esnault
U. Chunduri
Huawei Technologies
September 18, 2017

LISP for the Mobile Network
draft-farinacci-lisp-mobile-network-02

Abstract

This specification describes how the LISP architecture and protocols can be used in a LTE/5G mobile network to support session survivable EID mobility. A recommendation is provided to SDOs on how to integrate LISP into the mobile network.

Table of Contents

1. Introduction

The LISP architecture and protocols [RFC6830] introduces two new numbering spaces, Endpoint Identifiers (EIDs) and Routing Locators (RLOCs) which provide an architecture to build overlays on top of the underlying Internet. Mapping EIDs to RLOC-sets is accomplished with a Mapping Database System. By using a level of indirection for routing and addressing, separating an address identifier from its location can allow flexible and scalable mobility. By assigning EIDs to mobile devices and RLOCs to the network nodes that support such mobile devices, LISP can provide seamless mobility.

For a reading audience unfamiliar with LISP, a brief tutorial level document is available at [I-D.ietf-lisp-introduction].

This specification will describe how LISP can be used to provide layer-3 mobility within and across an LTE [LTE401-3GPP] [LTE402-3GPP] and 5G [ARCH5G-3GPP] [PROC5G-3GPP] mobile network.

The following are the design requirements:

1. Layer-3 address mobility is provided within a mobile network RAN supported by a pGW region (intra-pGW) as well as across pGW regions (inter-pGW).

2. UE nodes can get layer-3 address mobility when roaming off the mobile network to support Fixed Mobile Convergence [FMC].

3. Transport layer session survivability exists while roaming within, across, and off of the mobile network.

4. No address management is required when UEs roam. EID addresses are assigned to UEs at subscription time. EIDs can be reassigned when UE ownership changes.

5. The design will make efficient use of radio resources thereby not adding extra headers to packets that traverse the RAN.

6. The design can support IPv4 unicast and multicast packet delivery and will support IPv6 unicast and multicast packet delivery.

7. The design will allow use of both the GTP [GTPv1-3GPP] [GTPv2-3GPP] and LISP [I-D.ietf-lisp-rfc6830bis] data-planes while using the LISP control-plane and mapping system.

8. The design can be used for either 4G/LTE and 5G mobile networks and may be able to support interworking between the different mobile networks.

9. The LISP architecture provides a level of indirection for routing and addressing. From a mobile operator’s perspective, these mechanisms provide advantages and efficiencies for the URLLC, FMC, and mMTC use cases. See Section 2 for definitions and references of these use cases.

The goal of this specification is take advantage of LISP’s non-disruptive incremental deployment benefits. This can be achieved by changing the fewest number of components in the mobile network. The proposal suggests adding LISP functionality only to eNodeB and pGW nodes. There are no hardware or software changes to the UE devices or the RF-based RAN to realize this architecture. The LISP mapping database system is deployed as an addition to the mobile network and does not require any coordination with existing management and provisioning systems.

Similar ID Oriented Networking (ION) mechanisms for the 5G [ARCH5G-3GPP] [PROC5G-3GPP] mobile network are also being considered in other standards organizations such as ETSI [ETSI-NGP] and ITU

[ITU-IMT2020]. The NGMN Alliance describes Locator/ID separation an enabler to meet Key Performance Indicator Requirements [NGMN].

2. Definition of Terms

xTR: Is a LISP node in the network that runs the LISP control-plane and data-plane protocols according to [I-D.ietf-lisp-rfc6830bis] and [I-D.ietf-lisp-rfc6833bis]. A formal definition of an xTR can be found in [RFC6830]. In this specification, a LISP xTR is a node that runs the LISP control-plane with the GTP data-plane.

EID: Is an Endpoint Identifier. EIDs are assigned to UEs and other Internet nodes in LISP sites. A formal definition of an EID can be found in [RFC6830].

UE EID: A UE can be assigned an IPv4 and/or an IPv6 address either statically, or dynamically as is the procedure in the mobile network today. These IP addresses are known as LISP EIDs and are registered to the LISP mapping system. These EIDs are used as the source address in packets that the UE originates.

RLOC: Is an Routing Locator. RLOCs are assigned to eNodeBs and pGWs and other LISP xTRs in LISP sites. A formal definition of an RLOC can be found in [RFC6830].

Mapping System: Is the LISP mapping database system that stores EID-to-RLOC mappings. The mapping system is centralized for use and distributed to scale and secure deployment. LISP Map-Register messages are used to publish mappings and LISP Map-Requests messages are used to lookup mappings. LISP Map-Reply messages are used to return mappings. EID-records are used as lookup keys, and RLOC-records are returned as a result of the lookup. Details can be found in [RFC6833].

LISP Control-Plane: In this specification, a LISP xTR runs the LISP control-plane which originates, consumes, and processes Map-Request, Map-Register, Map-Reply, and Map-Notify messages.

RAN: Radio Access Network where UE nodes connect to eNodeB nodes via radios to get access to the Internet.

EPC: Evolved Packet Core [EPS-3GPP] system is the part of the mobile network that allows the RAN to connect to a data packet network. The EPC is a term used for the 4G/LTE mobile network.

NGC: Next Generation Core [EPS-3GPP] system is the part of the 5G mobile network that allows the RAN to connect to a data packet network.

GTP: GTP [GTPv1-3GPP] [GTPv2-3GPP] is the UDP tunneling mechanism used in the LTE/4G and 5G mobile network.

UE: User Equipment as defined by [GPRS-3GPP] which is typically a mobile phone. The UE is connected to the network across the RAN to eNodeB nodes.

eNodeB: Is the device defined by [GPRS-3GPP] which borders the RAN and connects UEs to the EPC in a 4G/LTE mobile network. The eNodeB nodes are termination point for a GTP tunnel and are LISP xTRs. The equivalent term in the 5G mobile network is "(R)AN" and "5G-NR", or simply "gNB". In this document, the two terms are used interchangeably.

pGW: Is the PDN-Gateway as defined by [GPRS-3GPP] connects the EPC in a 4G/LTE mobile network to the Internet. The pGW nodes are termination point for a GTP tunnel and is a LISP xTR. The equivalent user/data-plane term in the 5G mobile network is the "UPF", which also has the capability to chain network functions. In this document, the two terms are used interchangeably.

URLLC: Ultra-Reliable and Low-Latency provided by the 5G mobile network for the shortest path between UEs [NGMN].

FMC: Fixed Mobile Convergence [FMC] is a term used that allows a UE device to move to and from the mobile network. By assigning a fixed EID to a UE device, LISP supports transport layer continuity between the mobile network and a fixed infrastructure such as a WiFi network.

mMTC: Massive Machine-Type Services [mMTC] is a term used to refer to using the mobile network for large-scale deployment of Internet of Things (IoT) applications.

3. Design Overview

LISP will provide layer-3 address mobility based on the procedures in [I-D.ietf-lisp-eid-mobility] where the EID and RLOCs are not co-located. In this design, the EID is assigned to the UE device and the RLOC(s) are assigned to eNodeB nodes. So any packets going to a UE are always encapsulated to the eNodeB that associates with the UE. For data flow from the UE to any EIDs (or destinations to non-LISP sites) that are outside of the EPC, use the RLOCs of the pGW nodes so the pGW can send packets into the Internet core (unencapsulated).

The following procedures are used to incorporate LISP in the EPC:

- UEs are assigned EIDs. They usually never change. They identify the mobile device and are used for transport connections. If privacy for EIDs is desired, refer to details in [I-D.ietf-lisp-eid-anonymity].

- eNodeB nodes are LISP xTRs. They have GTP, and optionally LISP, tunnels to the pGW nodes. The eNodeB is the RLOC for all EIDs assigned to UE devices that are attached to the eNodeB.

- pGW nodes are LISP xTRs. They have GTP, and optionally LISP, tunnels to the eNodeB nodes. The pGW is the RLOC for all traffic destined for the Internet.

- The LISP mapping system runs in the EPC. It maps EIDs to RLOC-sets.

- Traffic from a UE to UE within a pGW region can be encapsulated from eNodeB to another eNodeB or via the pGW, acting as an RTR [RFC6830], to provide data-plane policy.

- Traffic from a UE to UE across a pGW region have these options for data flow:

    1. Encapsulation by a eNodeB in one region to a eNodeB in another region.

    2. Encapsulation by a eNodeB in one region to a pGW in the same region and then the pGW reencapsulates to a eNodeB in another region.

    3. Encapsulation by a eNodeB in one region to a pGW in another region and then the pGW reencapsulates to a eNodeB in its same region

- Note when encapsulation happens between a eNodeB and a pGW, GTP is used as the data-plane and when encapsulation between two eNodeBs occur, LISP can be used as the data-plane when there is no X2 interface [X2-3GPP] between the eNodeB nodes.

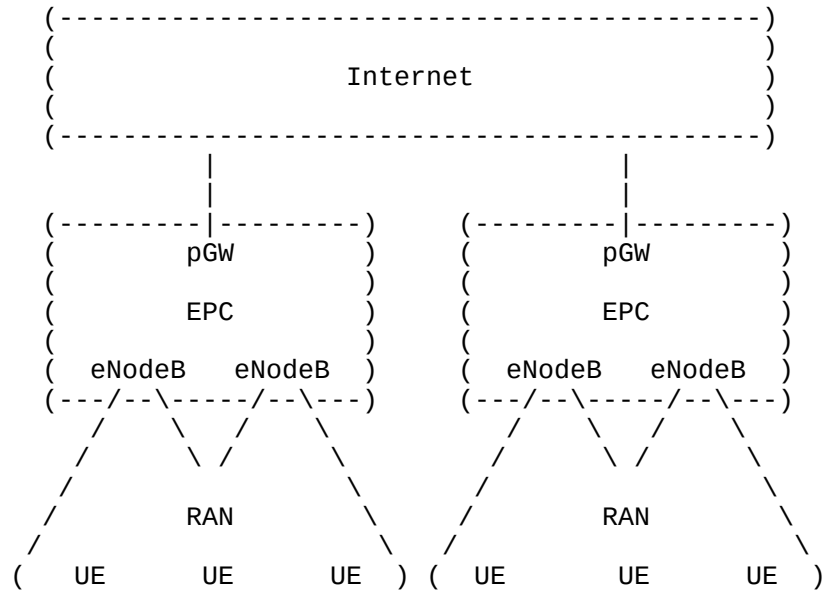
- The pGW nodes register their RLOCs for a default EID-prefix to the LISP mapping system. This is done so eNodeB nodes can find pGW nodes to encapsulate to.

- The eNodeB nodes register EIDs to the mapping system for the UE nodes. The registration occurs when eNodeB nodes discover the layer-3 addresses of the UEs that connect to them. The eNodeB nodes register multiple RLOCs associated with the EIDs to get multi-homing and path diversity benefits from the EPC network.

- When a UE moves off a eNodeB, the eNodeB node deregisters itself as an RLOC for the EID associated with the UE.

- Optionally, and for further study for future architectures, the eNodeB or pGW could encapsulate to an xTR that is outside of the EPC network. They could encapsulate to a LISP CPE router at a branch office, a LISP top-of-rack router in a data center, a LISP wifi access-point, LISP border routers at a hub site, and even a LISP router running in a VM or container on a server.

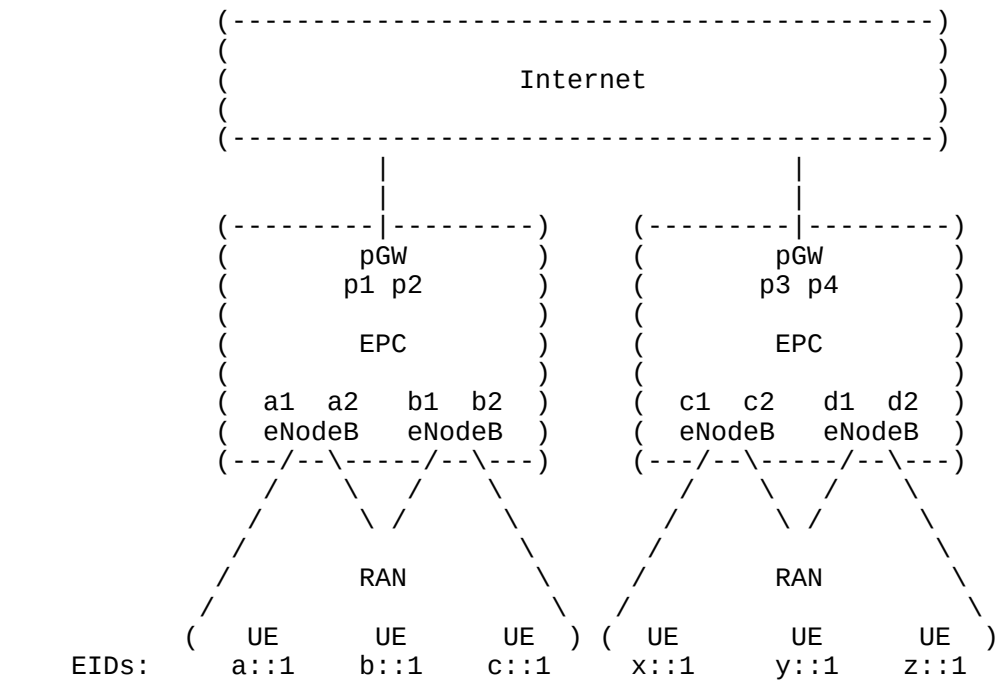
The following diagram illustrates the LTE mobile network topology and structure [LTE401-3GPP] [LTE402-3GPP]:



LTE/5G Mobile Network Architecture

The following diagram illustrates how LISP is used on the mobile network:

(1) IPv6 EIDs are assigned to UEs.
(2) RLOCs assigned to eNodeB nodes are [a1,a2], [b1,b2], [c1,c2], [d1,d2] on their uplink interfaces.
(3) RLOCs assigned to pGW nodes are [p1,p2], [p3,p4].
(4) RLOCs can be IPv4 or IPv6 addresses or mixed RLOC-sets.



Mobile Network with EID/RLOC Assignment

The following table lists the EID-to-RLOC entries that reside in the LISP Mapping System when the above UEs are are attached to the 4 eNodeBs:

EID-Record	RLOC-Record	Commentary	Footnote
0::/0	[p1,p2,p3 p4]	eNodeBs encap to p1-p4 for Internet destinations which are non-EIDs	(1)
a::1/128	[a1,a2]	pGWs load-split traffic to [a1,a2] for UE a::1 and it can move to [b1,b2]	(2)
b::1/128	[a1,a2]	eNodeB tracks both UEs a::1 and b::1, it can do local routing between the UEs	(3)
c::1/128	[b1,b2]	UE c::1 can roam to [c1,c2] or [d1,d2], may use pGW [p1,p2] after move	(4)
x::1/128	[c1,c2]	UE x::1 can talk directly to UE y::1, eNodeBs encap to each other	(5)
y::1/128	[d1,d2]	UE can talk to Internet when [d1,d2], encap to pGW [p3,p4] or use backup [p1,p2]	(6)
z::1/128	[d1,d2]	UE z::1 can talk to a::1 directly where [d1,d2] encaps to [a1,a2]	(7)

(1) For packets that flow from UE nodes to destinations that are not in LISP sites, the eNodeB node use one of the RLOCs p1, p2, p3, or p4 as the destination address in the outer encapsulated header. Encapsulated packets are then routed by the EPC core to the pGW nodes. In turn, the pGW nodes, then route packets into the Internet core.

(2) Packets that arrive to pGW nodes from the Internet destined to UE nodes are encapsulated to one of the eNodeB RLOCs a1, a2, b1, b2. When UE, with EID a::1 is attached to the leftmost eNodeB, the EID a::1 is registered to the mapping system with RLOCs a1 and a2. When UE with EID c::1 is attached to the rightmost eNodeB (in the left region), the EID c::1 is registered to the mapping system with RLOCs b1 and b2.

(3) If UE with EID a::1 and UE with EID b::1 are attached to the same eNodeB node, the eNodeB node tracks what radio interface to use to route packets from one UE to the other.

(4) If UE with EID c::1 roams away from eNodeB with RLOCs b1 and b2, to the eNodeB with RLOCs c1 and c2 (in the rightmost region), packets destined toward the Internet, can use any pGW. Any packets that flow back from the Internet can use any pGW. In either case, the pGW is

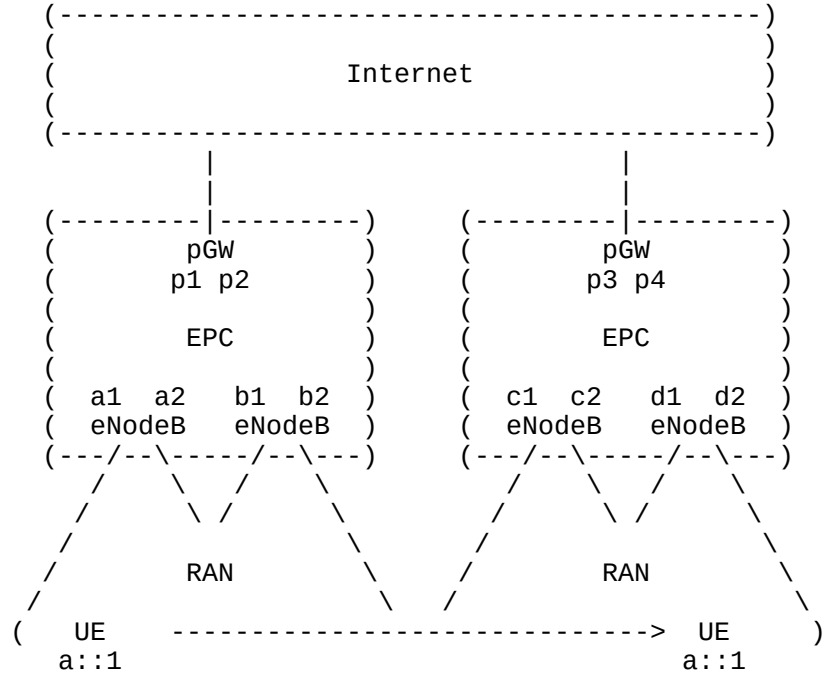
informed by the mapping system that the UE with EID c::1 has new RLOCs and should now encapsulate to either RLOC c1 or c2.

(5) When UE with EID x::1 is attached to eNodeB with RLOCs c1 and c2 and UE with EID y::1 is attached to eNodeB with RLOCs d1 and d2, they can talk directly, on the shortest path to each eNodeB, when each encapsulate packets to each other’s RLOCs.

(6) When packets from UE with EID y::1 are destined for the Internet, the eNodeB with RLOCs d1 and d2 that the UE is attached to can use any exit pGWs RLOCs p1, p2, p3, or p4.

(7) UE with EID z::1 can talk directory to UE with EID a::1 by each eNodeB they are attached to encapsulsates to each other’s RLOCs. In case (5), the two eNodeB’s were in the same region. In this case, the eNodeBs are in different regions.

The following abbreviated diagram shows a topology that illustrates how a UE roams with LISP across pGW regions:



UE EID Mobility

The contents of the LISP mapping database before UE moves:

EID-Record	RLOC-Record	Commentary
0::/0	[p1,p2,p3,p4]	eNodeB [a1,a2] encaps to p1-p4 for Internet destinations when a::1 on eNodeB [a1,a2]
a::1/128	[a1,a2]	Before UE moves to other pGW region

The contents of the LISP mapping database after UE moves:

EID-Record	RLOC-Record	Commentary
0::/0	[p1,p2,p3,p4]	eNodeB [d1,d2] encaps to p1-p4 for Internet destinations when a::1 moves to eNodeB [d1,d2]
a::1/128	[d1,d2]	After UE moves to new pGW region

4. Addressing and Routing

UE based EID addresses will be IPv6 addresses. It will be determined at a future time what length the IPv6 prefix will be to cover all UEs in a mobile network. This coarse IPv6 prefix is called an EID-prefix where more-specific EID-prefixes will be allocated out of it for each pGW node. Each pGW node is responsible for advertising the more-specific EID-prefix into the Internet routing system so they can attract packets from non-EIDs nodes to UE EIDs.

An RLOC address will either be an IPv4 or IPv6 address depending on the support for single or dual-stack address-family in the EPC network. An RLOC-set in the mapping system can have a mixed address-family locator set. There is no requirement for the EPC to change to support one address-family or the other. And there is no requirement for the EPC network to support IPv4 multicast or IPv6 multicast. The LISP overlay will support both.

The only requirement for RLOC addresses is that they are routable in the EPC and the Internet core network.

The requirements of the LISP and GTP data-plane overlay is to support a layer-3 overlay network only. There is no architectural requirement to support layer-2 overlays. However, operators may want to provide a layer-2 LAN service over their mobile network. Details about how LISP supports layer-2 overlays can be found in [I-D.ietf-lisp-eid-mobility].

5. eNodeB LISP Functionality

The eNodeB node runs as a LISP xTR for control-plane functionality and runs GTP for data-plane functionality. Optionally, the LISP data-plane can be used to establish dynamic tunnels from one eNodeB node to another eNodeB node.

The eNodeB LISP xTR will follow the procedures of [I-D.ietf-lisp-eid-mobility] to discover UE based EIDs, track them by monitoring liveness, registering them when appear, and deregistering them when they move away. Since the eNodeB node is an xTR, it is acting as a layer-3 router and the GTP tunnel from the eNodeB node to the pGW node is realizing a layer-3 overlay. This will provide scaling benefits since broadcast and link-local multicast packets won’t have to travel across the EPC to the pGW node.

A day in the life of a UE originated packet:

1. The UE node originates an IP packet over the RAN.

2. The eNodeB receives the packet, extracts the source address from the packet, learns the UE based EID, stores its RAN location locally and registers the EID to the mapping system.

3. The eNodeB extracts the destination address, looks up the address in the mapping system. The lookup returns the RLOC of a pGW node if the destination is not an EID or an RLOC eNodeB node if the destination is a UE based EID.

4. The eNodeB node encapsulates the packet to the RLOC using GTP or optionally the LISP data-plane.

It is important to note that in [I-D.ietf-lisp-eid-mobility], EID discovery occurs when a LISP xTR receives an IP or ARP/ND packet. However, if there are other methods to discover the EID of a device, like in UE call setup, the learning and registration referenced in Paragraph 2 can happen before any packet is sent.

6. pGW LISP Functionality

The pGW node runs as a LISP xTR for control-plane functionality and runs GTP for data-plane functionality. Optionally, the LISP data-plane can be used to establish dynamic tunnels from one pGW node to another pGW or eNodeB node.

The pGW LISP xTR does not follow the EID mobility procedures of [I-D.ietf-lisp-eid-mobility] since it is not responsible for discovering UE based EIDs. A pGW LISP xTR simply follows the procedures of a PxTR in [RFC6830] and for interworking to non-EID sites in [RFC6832].

A day in the life of a pGW received packet:

1. The pGW node receives a IP packet from the Internet core.

2. The pGW node extracts the destination address from the packet and looks it up in the LISP mapping system. The lookup returns an RLOC of a eNodeB node. Optionally, the RLOC could be another pGW node.

3. The pGW node encapsulates the packet to the RLOC using GTP or optionally the LISP data-plane.

7. Compatible Data-Plane using GTP

Since GTP is a UDP based encapsulating tunnel protocol, it has the same benefits as LISP encapsulation. At this time, there appears to

be no urgent need to not continue to use GTP for tunnels between a eNodeB nodes and between a eNodeB node and a pGW node.

There are differences between GTP tunneling and LISP tunneling. GTP tunnels are setup at call initiation time. LISP tunnels are dynamically encapsulating, used on demand, and don't need setup or teardown. The two tunneling mechanisms are a hard state versus soft state tradeoff.

This specification recommends for early phases of deployment, to use GTP as the data-plane so a transition for it to use the LISP control-plane can be achieved more easily. At later phases, the LISP data-plane may be considered so a more dynamic way of using tunnels can be achieved to support URLLC.

This specification recommends the use of procedures from [I-D.ietf-lisp-eid-mobility] and NOT the use of LISP-MN [I-D.ietf-lisp-mn]. Using LISP-MN states that a LISP xTR reside on the mobile UE. This is to be avoided so extra encapsulation header overhead is NOT sent on the RAN. The LISP data-plane or control-plane will not run on the UE.

8. Roaming and Packet Loss

Using LISP for the data-plane has some advantages in terms of providing near-zero packet loss. In the current mobile network, packets are queued on the eNodeB node the UE is roaming to or rerouted on the eNodeB node the UE has left. In the LISP architecture, packets can be sent to multiple "roamed-from" and "roamed-to" nodes while the UE is moving or is off the RAN. See mechanisms in [I-D.ietf-lisp-predictive-rlocs] for details.

9. Mobile Network LISP Mapping System

The LISP mapping system stores and maintains EID-to-RLOC mappings. There are two mapping database transport systems that are available for scale, LISP-ALT [RFC6836] and LISP-DDT [RFC8111]. The mapping system will store EIDs assigned to UE nodes and the associated RLOCs assigned to eNodeB nodes and pGW nodes. The RLOC addresses are routable addresses by the EPC network.

This specification recommends the use of LISP-DDT.

10. Multicast Considerations

Since the mobile network runs the LISP control-plane, and the mapping system is available to support EIDs for unicast packet flow, it can

also support multicast packet flow. Support for multicast can be provided by the LISP/GTP overlay with no changes to the EPC network.

Multicast (S-EID,G) entries can be stored and maintained in the same mapping database that is used to store UE based EIDs. Both Internet connected nodes, as well as UE nodes, can source multicast packets. The protocol procedures from [I-D.ietf-lisp-signal-free-multicast] are followed to make multicast delivery available. Both multicast packet flow and UE mobility can occur at the same time.

A day in the life of a 1-to-many multicast packet:

1. A UE node joins an (S,G) multicast flow by using IGMPv2 or IGMPv3.

2. The eNodeB node records which UE on the RAN should get packets sourced by S and destined for group G.

3. The eNodeB node registers the (S,G) entry to the mapping system with its RLOC according to the receiver site procedures in [I-D.ietf-lisp-signal-free-multicast]. The eNodeB does this to show interest in joining the multicast flow.

4. When other UE nodes join the same (S,G), their associated eNodeB nodes will follow the procedures in steps 1 through 3.

5. The (S,G) entry stored in the mapping database has an RLOC-set which contains a replication list of all the eNodeB RLOCs that registered.

6. A multicast packet from source S to destination group G arrives at the pGW. The pGW node looks up (S,G), gets returned the replication list of all joined eNodeB nodes and replicates the multicast packet by encapsulating the packet to each of them.

7. Each eNodeB node decapsulates the packet and delivers the multicast packet to one or more IGMP-joined UEs on the RAN.

11. Security Considerations

For control-plane authentication and authorization procedures, this specification recommends the mechanisms in [I-D.ietf-lisp-rfc6833bis], LISP-SEC [I-D.ietf-lisp-sec] AND LISP-ECDSA [I-D.farinacci-lisp-ecdsa-auth].

For data-plane privacy procedures, this specification recommends the mechanisms in [RFC8061] When the LISP data-plane is used. otherwise, the EPC must provide data-plane encryption support.

12. IANA Considerations

There are no specific requests for IANA.

13. SDO Recommendations

The authors request other Standards Development Organizations to consider LISP as a technology for device mobility. It is recommended to start with this specification as a basis for design and develop more deployment details in the appropriate Standards Organizations. The authors are willing to facilitate this activity.

14. References

14.1. Normative References

[RFC1700] Reynolds, J. and J. Postel, "Assigned Numbers", RFC 1700, DOI 10.17487/RFC1700, October 1994, <https://www.rfc-editor.org/info/rfc1700>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <https://www.rfc-editor.org/info/rfc6830>.

[RFC6832] Lewis, D., Meyer, D., Farinacci, D., and V. Fuller, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, DOI 10.17487/RFC6832, January 2013, <https://www.rfc-editor.org/info/rfc6832>.

[RFC6833] Fuller, V. and D. Farinacci, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, DOI 10.17487/RFC6833, January 2013, <https://www.rfc-editor.org/info/rfc6833>.

[RFC6836] Fuller, V., Farinacci, D., Meyer, D., and D. Lewis, "Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, DOI 10.17487/RFC6836, January 2013, <https://www.rfc-editor.org/info/rfc6836>.

[RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <https://www.rfc-editor.org/info/rfc8060>.

[RFC8061] Farinacci, D. and B. Weis, "Locator/ID Separation Protocol (LISP) Data-Plane Confidentiality", RFC 8061, DOI 10.17487/RFC8061, February 2017, <https://www.rfc-editor.org/info/rfc8061>.

[RFC8111] Fuller, V., Lewis, D., Ermagan, V., Jain, A., and A. Smirnov, "Locator/ID Separation Protocol Delegated Database Tree (LISP-DDT)", RFC 8111, DOI 10.17487/RFC8111, May 2017, <https://www.rfc-editor.org/info/rfc8111>.

14.2. Informative References

[ARCH5G-3GPP]
3GPP, "System Architecture for the 5G System", TS.23.501 https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144, December 2016.

[EPS-3GPP]
3GPP, "Non-Access-Stratum (NAS) Protocol for Evolved Packet System (EPS); Stage 3", TS.23.501 https://portal.3gpp.org/desktopmodules/specifications/specificationdetails.aspx?specificationid=1072, January 2015.

[ETSI-NGP]
ETSI-NGP, "NGP Evolved Architecture for mobility using Identity Oriented Networks", NGP-004, version 0.0.3 https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=50531, May 2017.

[FMC] ipv6.com, "FIXED MOBILE CONVERGENCE", https://www.ipv6.com/mobile/fixed-mobile-convergence/, November 2006.

[GPRS-3GPP]
3GPP, "General Packet Radio Service (GPRS) for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) Access", TS23.401 Release 8 https://portal.3gpp.org/desktopmodules/specifications/specificationdetails.aspx?specificationid=849, January 2015.

[GTPv1-3GPP]
3GPP, "General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)", TS.29.281 https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1699, January 2015.

[GTPv2-3GPP]
3GPP, "3GPP Evolved Packet System (EPS); Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C); Stage 3", TS.29.274 https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1692, January 2015.

[I-D.farinacci-lisp-ecdsa-auth]
Farinacci, D. and E. Nordmark, "LISP Control-Plane ECDSA Authentication and Authorization", draft-farinacci-lisp-ecdsa-auth-00 (work in progress), July 2017.

[I-D.ietf-lisp-eid-anonymity]
Farinacci, D., Pillay-Esnault, P., and W. Haddad, "LISP EID Anonymity", draft-ietf-lisp-eid-anonymity-00 (work in progress), August 2017.

[I-D.ietf-lisp-eid-mobility]
Portoles-Comeras, M., Ashtaputre, V., Moreno, V., Maino, F., and D. Farinacci, "LISP L2/L3 EID Mobility Using a Unified Control Plane", draft-ietf-lisp-eid-mobility-00 (work in progress), May 2017.

[I-D.ietf-lisp-introduction]
Cabellos-Aparicio, A. and D. Saucez, "An Architectural Introduction to the Locator/ID Separation Protocol (LISP)", draft-ietf-lisp-introduction-13 (work in progress), April 2015.

[I-D.ietf-lisp-mn]
Farinacci, D., Lewis, D., Meyer, D., and C. White, "LISP Mobile Node", draft-ietf-lisp-mn-00 (work in progress), April 2017.

[I-D.ietf-lisp-predictive-rlocs]
Farinacci, D. and P. Pillay-Esnault, "LISP Predictive RLOCs", draft-ietf-lisp-predictive-rlocs-00 (work in progress), June 2017.

[I-D.ietf-lisp-rfc6830bis]
Farinacci, D., Fuller, V., Meyer, D., Lewis, D., and A. Cabellos-Aparicio, "The Locator/ID Separation Protocol (LISP)", draft-ietf-lisp-rfc6830bis-05 (work in progress), August 2017.

[I-D.ietf-lisp-rfc6833bis]
Fuller, V., Farinacci, D., and A. Cabellos-Aparicio, "Locator/ID Separation Protocol (LISP) Control-Plane", draft-ietf-lisp-rfc6833bis-05 (work in progress), May 2017.

[I-D.ietf-lisp-sec]
Maino, F., Ermagan, V., Cabellos-Aparicio, A., and D. Saucez, "LISP-Security (LISP-SEC)", draft-ietf-lisp-sec-12 (work in progress), November 2016.

[I-D.ietf-lisp-signal-free-multicast]
Moreno, V. and D. Farinacci, "Signal-Free LISP Multicast", draft-ietf-lisp-signal-free-multicast-06 (work in progress), August 2017.

[ITU-IMT2020]
ITU-FG, "Focus Group on IMT-2020", https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.687-2-199702-I!!PDF-E.pdf.

[LTE401-3GPP]
3GPP, "General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access", TS.23.401 https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=849, January 2015.

[LTE402-3GPP]
3GPP, "Architecture enhancements for non-3GPP accesses", TS.23.402 https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=850, January 2015.

[mMTC]
NGMN Alliance, "NGMN KPIs and Deployment Scenarios for Consideration for IMT2020", https://www.ngmn.org/uploads/media/151204_NGMN_KPIs_and_Deployment_Scenarios_for_Consideration_for_IMT_2020_-_LS_Annex_V1_approved.pdf, December 2015.

[NGMN] NGMN Alliance, "5G End-to-End Architecture Framework", NGMN https://www.ngmn.org/uploads/media/170511_NGMN_E2EArchFramework_v0.6.5.pdf, March 2016.

[PROC5G-3GPP] 3GPP, "Procedures for the 5G System", TS.23.502 https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3145, December 2016.

[X2-3GPP] 3GPP, "Evolved Universal Terrestrial Radio Access Network (E-UTRAN); X2 Application Protocol (X2AP)", TS.36.423 https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2452, June 2017.

Appendix A. Acknowledgments

The authors would like to thank Gerry Foster and Peter Ashwood Smith for their expertise with 3GPP mobile networks and for their early review and contributions. The authors would also like to thank Fabio Maino, Malcolm Smith, and Marc Portoles for their expertise in both 5G and LISP as well as for their early review comments.

The authors would like to give a special thank you to Ryosuke Kurebayashi from NTT Docomo for his operational and practical commentary.

Appendix B. Document Change Log

B.1. Changes to draft-farinacci-lisp-mobile-network-02.txt

- Posted mid September 2017.
- Editorial fixes from draft -01.

B.2. Changes to draft-farinacci-lisp-mobile-network-01.txt

- Posted September 2017.
- Explain each EID case illustrated in the "Mobile Network with EID/RLOC Assignment" diagram.
- Make a reference to mMTC as a 3GPP use-case for 5G.
- Add to the requirements section how mobile operators believe that using Locator/ID separation mechanisms provide for more efficient mobile netwowks.

- Indicate that L2-overlays is not recommended by this specification as the LISP mobile network architeture but how operators may want to deploy a layer-2 overlay service.

B.3. Changes to draft-farinacci-lisp-mobile-network-00.txt

- Initial draft posted August 2017.

Authors' Addresses

Dino Farinacci
lispers.net
San Jose, CA
USA

Email: farinacci@gmail.com

Padma Pillay-Esnault
Huawei Technologies
Santa Clara, CA
USA

Email: padma@huawei.com

Uma Chunduri
Huawei Technologies
Santa Clara, CA
USA

Email: uma.chunduri@huawei.com

DMM WG
Internet-Draft
Intended status: Standards Track
Expires: August 28, 2018

S. Gundavelli
Cisco
M. Liebsch
NEC
S. Matsushima
SoftBank

February 24, 2018

Mobility-aware Floating Anchor (MFA)
draft-gundavelli-dmm-mfa-00.txt

Abstract

IP mobility protocols are designed to allow a mobile node to remain reachable while moving around in the network. The currently deployed mobility management protocols are anchor-based approaches, where a mobile node's IP sessions are anchored on a central node. The mobile node's IP traffic enters and exits from this anchor node and it remains as the control point for all subscriber services. This architecture based on fixed IP anchors comes with some complexity and there is some interest from the mobile operators to eliminate the use of fixed anchors, and other residual elements such as the overlay tunneling that come with it.

This document describes a new approach for realizing a mobile user-plane that does not require fixed IP anchors. The architectural-basis for this approach is the separation of control and user plane, and the use of programmability constructs of the user-plane for traffic steering. This approach is referred to as, Mobility-aware Floating Anchor (MFA).

Status of this Memo

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

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

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

This Internet-Draft will expire on August 28, 2018.

Table of Contents

1. Introduction

IP mobility protocols are designed to allow a mobile node to remain reachable while moving around in the network. The currently deployed mobility management protocols are anchor-based approaches, where a mobile node's IP sessions are anchored on a central node. The mobile node's IP traffic enters and exits from this anchor node and it remains as the control point for all subscriber services. This architecture based on fixed IP anchors comes with some complexity and there is some interest from the mobile operators to eliminate the use of fixed anchors, and other residual elements such as the overlay tunneling that come with it. Some of the key objectives for this effort are listed below.

- Access-agnostic, shared user-plane that can be used for multiple access technologies
- Optimized Routing for the mobile node's IP flows with topology awareness and leveraging the transport QoS
- Elimination of overlay tunnels from the user-plane network for avoiding packet fragmentation, and reducing encapsulation related packet-size overhead
- Elimination of centralized mobility anchors and shift towards a distributed mobility architecture, leveraging the edge compute at radio-access network for offloading some of the subscriber management services
- Co-existence with control-plane and user-plane separated architecture; a stateless user-plane with no tunnels, and a control plane with the business/service logic
- Support for services including accounting, charging, lawful-interception and other user plane services

Currently, there is a study item in 3GPP to explore options for simplifying the mobile user-plane. There are few proposals in IETF, which are presented as candidate solutions for user-plane simplification. However, each of these proposals come with certain complexity and do not leverage the 3GPP control plane, or the programmability aspects of the user-plane. For example, ILA defines a translation scheme without the need for overlay tunnels, but it also introduces significant amount of translation related state in the user-plane, and additionally introduces a new control-plane protocol for managing the mapping tables and the cache states. Therefore, we believe that none of the currently known approaches can adequately meet the stated goals for user-plane simplification.

This document describes a new approach for realizing a mobile user-plane that does not require any fixed IP anchors. The first-hop router on the link where the mobile node is attached remains as the IP anchor and thereby eliminating the need for IP tunneling to some central anchor node. Even when the mobile node moves in the network and changes its point of attachment, the IP anchor is always the first-hop router on that new link. The MFA entities will track the mobile node's movements in the network and will ensure the mobile node's IP flows always take the most optimal routing path. This is achieved by MFA entities programming the needed traffic steering rules for moving mobile node's IP packets directly between the correspondent node and the mobile node's edge anchor, which can be relocated to a new edge, e.g. in case of mobility. Furthermore, this approach does not require a new control-plane protocol, but instead leverages the SDN interfaces of the user-plane, and the mobility events in the control-plane for managing IP mobility. The architectural basis for this approach is the separation of control and user plane, and the use of programmability constructs of the user-plane for traffic steering. This approach is referred to as, Mobility-aware Floating Anchor (MFA). The rest of the document explains the operational details of the MFA approach.

2. Conventions and Terminology

2.1. Conventions

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

2.2. Terminology

All mobility related terms used in this document are to be interpreted as defined in the IETF mobility specifications, including [RFC5213] and [RFC6275]. Additionally, this document uses the following terms:

MFA Domain

MFA domain refers to the network where the mobility management of a mobile node is handled by the MFA entities. The MFA domain includes MFA mobile node anchors, MFA corresponding node anchors, and MFA node controller, between which security associations can be set up for authorizing the configuration of traffic steering policies and other mobility management functions.

MFA Mobile Node Anchor (MFA-MNA)

Its an MFA function located in the user-plane network very close to the layer-2 access-point to where the mobile node is attached. It is typically on the first-hop router for the mobile node's IP traffic. The node hosting this function is required to support the standard IPv6 packet forwarding function, FPC or a similar interface for policy configuration, and packet steering functions such as based on SRv6 or alternative means that can support per-flow or per-flow-aggregate traffic steering. Typically, the MFA-MNA function will be collocated with the User Plane Function (UPF) in the 3GPP 5G system architecture.

MFA Corresponding Node Anchor (MFA-CNA)

Its an MFA function located in the user-plane node in the path between the mobile node and the correspondent node. If the correspondent node is another mobile node in the MFA domain, then the MFA-CNA is on the first hop router on the link shared with the correspondent node. The node hosting this function is required to support the standard IPv6 packet forwarding function, FPC or a similar interface for policy configuration, and packet steering functions such as based on SRv6 or alternative means that can support per-flow or per-flow-aggregate traffic steering. Typically, the MFA-CNA function will be collocated with the IP forwarding nodes on the N6 interface of the 3GPP 5G system architecture.

MFA Node

A generic term used for referring to MFA-MNA, or the MFA-CNA.

MFA Node-Controller (MFA-NC)

The is the function that controls the forwarding policies on the MFA-MNA and MFA-CNA nodes. This entity interfaces with the MFA node using the FPC interface [I-D.ietf-dmm-fpc-cpdp], or a similar interface that support user-plane policy configuration. This is typically co-located with the SMF, or the AMF functions in the 3GPP 5G system architecture, and on WLAN controller in the case of Wi-Fi access architectures.

Node Location Database (NLDB)

A database that contains the location information of every mobile node that is part of the MFA domain and is currently attached to the network.

Network Topology Database (NTDB)

A database that contains the MFA node information along with the link state and directly connected neighbor information.

Home Network Prefix (HNP)

An IPv6 prefix assigned to the mobile node. This prefix is hosted by the MFA-MNA on the access link shared with the mobile node. The network will provide mobility support for the HNP prefixes. A meta-data tag indicating the mobility property [I-D.ietf-dmm-ondemand-mobility] is included in router advertisements and in address assignment related protocol messages.

Local Network Prefix (LNP)

An IPv6 prefix assigned to the mobile node. This prefix is hosted by the MFA-MNA on the access link shared with the mobile node. The network will not provide mobility support for the LNP prefixes. A meta-data tag indicating that there is no mobility support [I-D.ietf-dmm-ondemand-mobility] is included in router advertisements and in address assignment related protocol messages.

3. Overview

This specification describes the MFA protocol. The MFA protocol is designed for providing mobility management support to a mobile node without the need for a fixed IP anchor. In this approach the mobile node's IP session is always anchored on the first-hop router sharing the link with the mobile node. The entities in the MFA domain track the mobile node's movements in the MFA domain and will provision the forwarding states in the user-plane nodes for optimal routing and for ensuring the anchor is always the first-hop router. Any time the mobile node moves within the MFA domain and resulting in the mobile node's IP flows going through the previous anchor, the mobility entities detect this event and a corrective action is taken by provisioning the forwarding nodes with the path stitching rules. The result of this approach is an user-plane with no fixed anchors, and dynamically programmed user-plane for mobility and optimal packet routing.

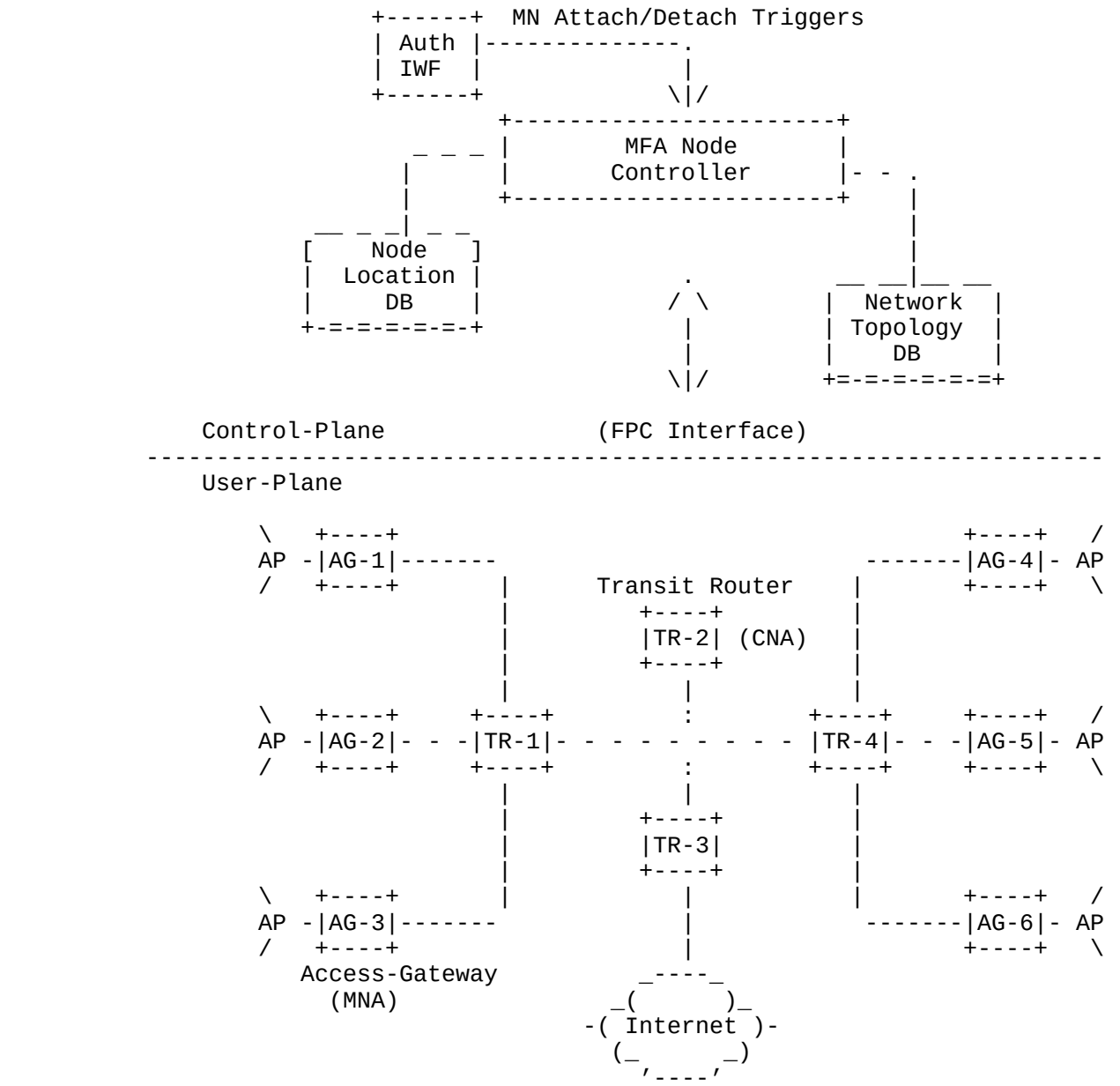
The following are the key functional entities in the MFA domain:

- MFA Node Controller (MFA-NC)
- MFA Mobile Node Anchor (MFA-MNA)

- MFA Correspondent Node anchor (MFA-CNA)

The MFA-NC is typically collocated with the access network specific control-plane functions. It interfaces with the radio network/ authentication functions for detecting the mobile node's movements in the MFA domain for managing the forwarding states in the user-plane entities, MFA-MNA and MFA-CNA. The MFA node controller requires access to node location database and network topology database. These are the conceptual entities that can be realized using existing elements that are already present in different access architectures.

The MFA-MNA and the MFA-CNA are the functions in the user-plane network and they are collocated with the elements in the network that perform IP packet forwarding functions. The MFA-MNA is typically located on the first-hop router and whereas the MFA-CNA can be collocated with the access-gateways and transit routers. These entities interface with the MNA-NC using FPC ([I-D.ietf-dmm-fpc-cpdp]), or an alternative interface), for managing the IP forwarding policies.



* MFA-MNA is collocated with the access gateways
** MFA-CNA is collocated with the access gateways and transit routers

Figure 1: Example of a MFA Domain

3.1. The Network Topology Database

The network topology database contains the complete and the current information about all the MFA nodes in the network. The information includes the capabilities of each node, supported functions, supported interfaces with the interface-type, connected neighbors, hosted prefixes on each link, security configuration and other related configuration elements. The topology database can be used to determine the route between two nodes within the MFA domain, or the best exit gateway for reaching a correspondent node outside the MFA domain.

3.2. The Node Location Database

The node location database consists of location information of each mobile node that is currently attached to the MFA domain. It also includes the type of attachment, previous anchor, and other information elements, such as the mobile node's connection status and detailed or approximate location (e.g. tracking area) in case of device dormancy. Typically, the MFA entities obtain this information from the control-plane functions in the access network. For example, a WLAN controller and the authentication functions will be able to provide this information in IEEE 802.11 based networks. In 5G system architecture this information can be obtained from AMF/SMF functions.

Below diagram is an example NLDB database.

MN Identifier	Current Anchor	Previous Anchor	Handover Type
MN1@ietf.org	AG1	-	NEW_ATTACH
MN2@ietf.org	AG6	AG2	HANDOVER
MN3@ietf.org	-	AG4	UNKNOWN

Figure 2: Example NLDB Table

3.3. Determination of the Correspondent Node Anchor

The anchor for a correspondent node is a MFA node that is closest to the correspondent node and is in path for all the MN-CN IP traffic flows. The MFA node controller leverages the topology database for the CN-anchor determination.

If the correspondent node is another mobile node in the MFA domain, then the CN-Anchor for that correspondent node is the access gateway to which it is currently attached.

If the correspondent node is outside the MFA domain, then the CN-anchor is typically the exit gateway, or any MFA node that is always in path for reaching the CN's network. This is typically the PE router of the data center that hosts the correspondent node service, or a programmable data plane node inside the data center.

The below illustration is an example topology of a MFA domain. The domain consists of MFA nodes, mobile and correspondent nodes. A query for CN2's anchor should result in finding AG4, as that is the MFA node in the traffic path and closest to CN2. Similarly, the query for CN3's anchor which is outside the MFA domain should result in finding TR3 as that is the last exit gateway in the MFA domain and closest to the CN3.

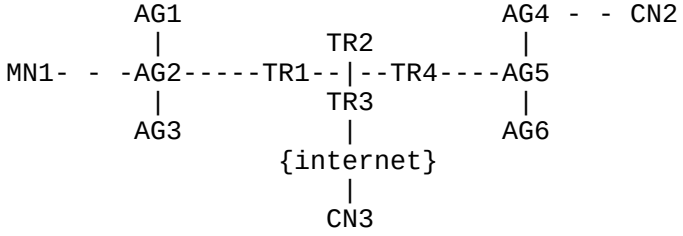


Figure 3: CN Anchor Determination - Example Topology

3.4. Traffic Steering Approaches

The MFA nodes support traffic steering approaches for moving the mobile node's IP traffic between the MFA nodes over the most optimal routing path. Segment Routing for IPv6 (SRv6) is one approach that this specification focuses on for steering the traffic between two points in the network, whereas the MFA-NC can utilize the available information from Network Topology- and Node Location Database to enforce policies in the MFA nodes in support of alternative data

plane protocols to enable traffic steering. Future versions of the document may include information about additional mechanisms.

When using SRv6 for traffic steering, the approaches specified in [I-D.ietf-dmm-srv6-mobile-uplane] and [I-D.filsfils-spring-srv6-network-programming] will be leveraged for moving the mobile node's IP traffic between the MFA-MNA and the MFA-CNA nodes. The SRv6 policy including the SID information and the associated functions are pushed from the MFA Node controller to the MFA nodes. This document mostly leverages the functions specified in those documents, but may require some changes to the SRv6 functions for reporting the flow meta-data of the non-optimal traffic flows to the MFA node controller. The definitions of those SRv6 functions will be specified in either in the future revisions of this document, or in other IETF documents.

The following table captures the possible SRv6 function activation when IP traffic steering approach is in use. This is only an example.

FLOW DIRECTION	MN-Anchor	CN-Anchor
MN to CN	Variant of T.Insert (Transit with insertion of SRv6 policy and may require trigger to MFA-NC such as activation of Flow.Report)	Variant of End.X (Or, End.B6, instantiation of a binding SID); Or, End.T for internet traffic
CN to MN	Variant of End.X (Layer-3 cross connect (Or, End.B6, instantiation of a binding SID	Variant of T.Insert (Transit with insertion of SRv6 policy and may require trigger to MFA-NC such as activation of Flow.Report.

Figure 4: Using SRv6 for Traffic Steering - Example

3.5. Mobile Node Attachment Triggers

The MFA domain relies on the access network for certain key events related to the mobile node's movements in the network. These events include:

- INITIAL_ATTACH - Initial Attachment of the mobile node to the MFA domain
- HANDOVER - Layer-2/Layer-3 Handover of the mobile node within the MFA Domain
- DETACH - Detachment of the mobile node from the MFA domain
- UNKNOWN - State of the mobile node is Unknown; TBD

The MFA node controller interfaces with the radio network and the authentication infrastructure for these events. These events drive the policy configuration on the MFA nodes.

3.6. Programming the User-plane

The MFA-NC leverages suitable southbound semantics and operation to enforce traffic steering rules in the selected access gateways (AG) and/or transient routers (TR). One suitable data model and operation is being specified in [I-D.ietf-dmm-fpc-cpdp] for Forwarding Policy Configuration (FPC). The model and operation applies in between a FPC Client function and an FPC Agent function.

A deployment of FPC with the specification per this document about MFA, the FPC Client is co-located with the MFA-NC, whereas the FPC Agent function is co-located with functions that enforce user plane configuration per the rules received from the FPC Client. The FPC Agent can either reside on an transport network- or SDN controller and be in charge of the configuration of multiple user plane nodes (MFA-TR, MFA-MA, MFA-CA), or an FPC Agent resides on each MFA node.

The following figure schematically draws an example how FPC can integrate with the functional MFA architecture per this specification. The example assumes that MFA nodes can be programmatically configured by an SDN Controller. Details about whether a single or multiple distributed SDN Controllers are deployed are left out.

The FPC data model includes the following components:

Data Plane Nodes (DPN) Model:

Representation of nodes in the data plane which can be selected and enforce rules per the control plane's directives. DPNs take a particular role, which is identified in the model. In the context of this document, the role of a DPN can be, for example, an anchor node or a transit router.

Topology Model:

Representation of DPNs in the network and associate in between DPNs. The FPC Client and Agent use the Topology to select most appropriate data plane node resources for a communication. In the context of this document, Topology has can be leveraged to implement the NTDB for the selection of steering paths and associated DPNs which function as MFA-MNA, MFA-CNA, or MFA-TR.

Policy Model:

Defines and identifies rules for enforcement at DPNs.

Mobility-Context:

Holds information associated with a mobile node and its mobility sessions. In the context of this document, Mobility-Context can be enriched with traffic steering related rules.

Monitor:

Provides mechanisms to register monitors (traffic, events) in the data plane and define status reporting schedules, which can be periodic or event-based. In the context of this document, Monitors may be used to detect traffic from a CN to an MN on an MFA node, which could result in a notification to the MFA-NC for path optimization and associated steering of traffic to the MN's current MFA-MNA.

Please refer to [I-D.ietf-dmm-fpc-cpdp] for model and operational details.

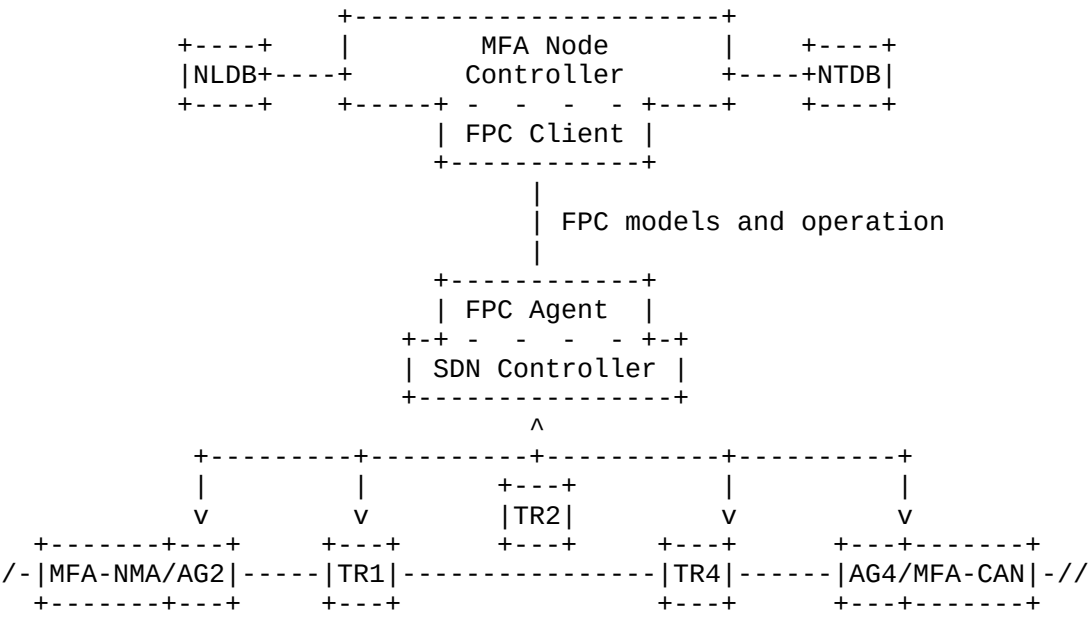


Figure 5: Deployment of the FPC models and operation in between the MFA-NC and MFA nodes on the user plane

4. Life of a Mobile Node in a MFA Domain

Reference Topology

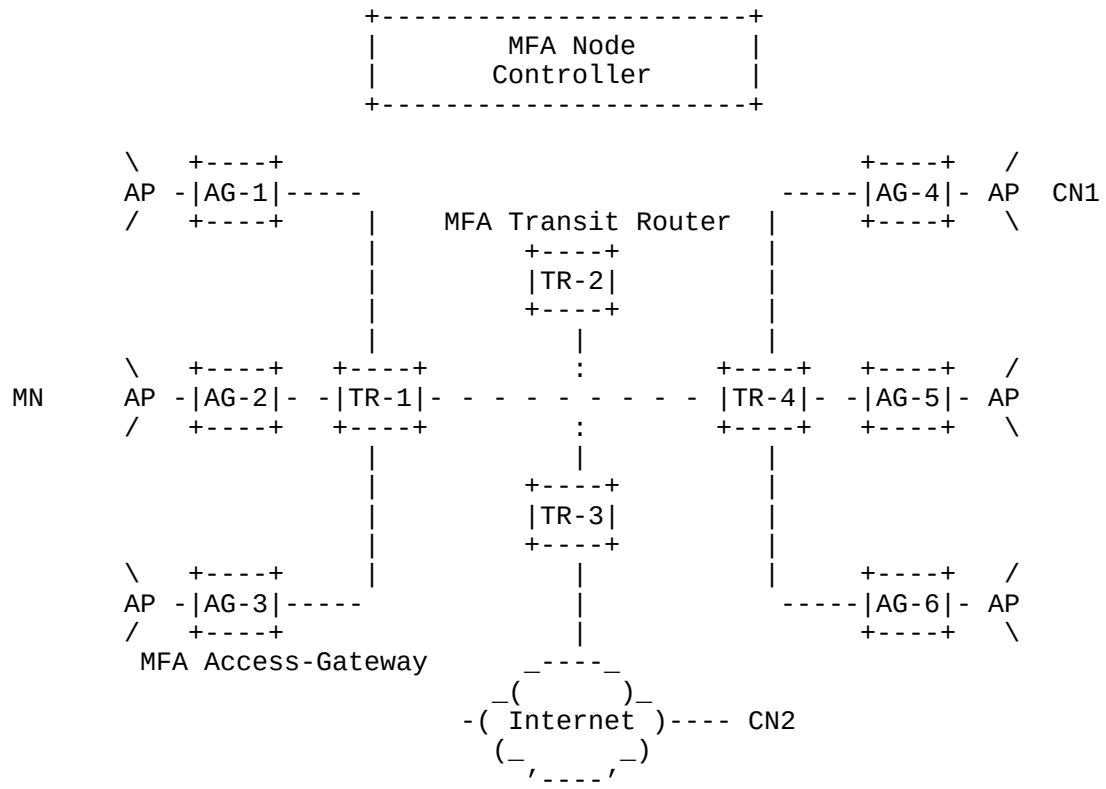


Figure 6: Reference Topology

4.1. MN's Initial Attachment to a MFA Domain

A mobile node, MN enters the MFA domain and attaches to the access point on the gateway AG-2.

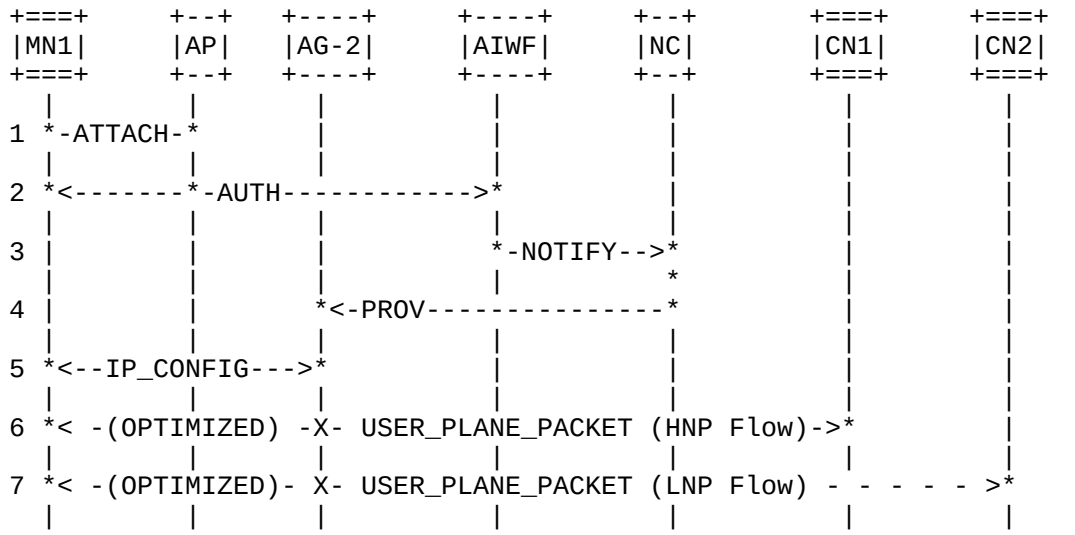


Figure 7: Mobile Node's Initial Attachment to a MFA Domain

- 1-ATTACH: The mobile node with NAI (MN1@ietf.org) performs a layer-2 attach to the access point. This access point is connected to the access-gateway, AG-2, over a layer-2 link. The mobile node anchor function is supported on AG-2 and is active.

- 2-AUTH: The mobile node completes the access authentication access technology specific access mechanisms. The mobile node's identity is established and is authorized for MFA domain access. The Authentication interworking (AUTH-IWK) function records the mobile node's identity, type of attach as INITIAL_ATTACH, and the current location of the mobile node in the access-network, to the node location database.

- 3-NOTIFY: The Auth-IWK function delivers the attach event to the MFA node controller. The information elements that are delivered include the mobile node identifier (MN-1@ietf.org), type of attach as INITIAL_ATTACH, and the identity of the access gateway, which is AG-2.

- 4-PROV: The NC provisions AG-2 for hosting the MN's home-network prefix(es). The assigned prefixes are HNP, H1::/64 and LNP, L1::/64. These prefixes are from a larger aggregate block (Ex: H1:://48; L1::/48) which are topologically anchored on AG-2. The policies for hosting the HNP prefixes on the link are provisioned using FPC interface. The AG-2 will include meta-data in the IPv6 RA messages for indicating the properties of the prefixes; H1::/64

as the prefix with mobility support and L1 as the prefix with no mobility support.

- o 5-IP_CONFIG: The mobile node generates one ore more IPv6 addresses using the prefixes H1 and L1. The generated addresses are tagged with the property meta-data in the host's source address policy table. This allows the applications on the mobile node to pick the addresses based on the application's mobility requirements.

- o 6-USER_PLANE_PACKET: The mobile node establishes IP flow with CN1. The source address is based on the prefix H1. This IP address will have mobility support. The packets associated with this flow will take the optimized routing path. There are no tunnels, or special traffic steering rules in the network.

- o 7-USER_PLANE_PACKET: The mobile node establishes IP flow with CN2. The source address is based on the prefix L1. This IP address will not have mobility support. There are no tunnels, or special traffic steering rules in the network.

4.2. MN's Roaming within the MFA Domain

The mobile node roams and changes its point of attachment. It was initially attached to the access network on AG-2 and now it attaches to access network on AG-6. At the time of roaming, the mobile node had two active IPv6 prefixes HNP, H1::/64 and LNP, L1::/64 and there were two active IP flows, one to CN1 using an IPv6 address from the prefix H1::/64 and another flow to CN2 using an IPv6 address from the prefix L1:://64. The MFA network will ensure the prefix H1::/64 will be routable on the new network and the active flow to CN1 will survive, however the prefix L1::/64 will not be routable in the new access network and therefore the flow to CN2 will not survive.

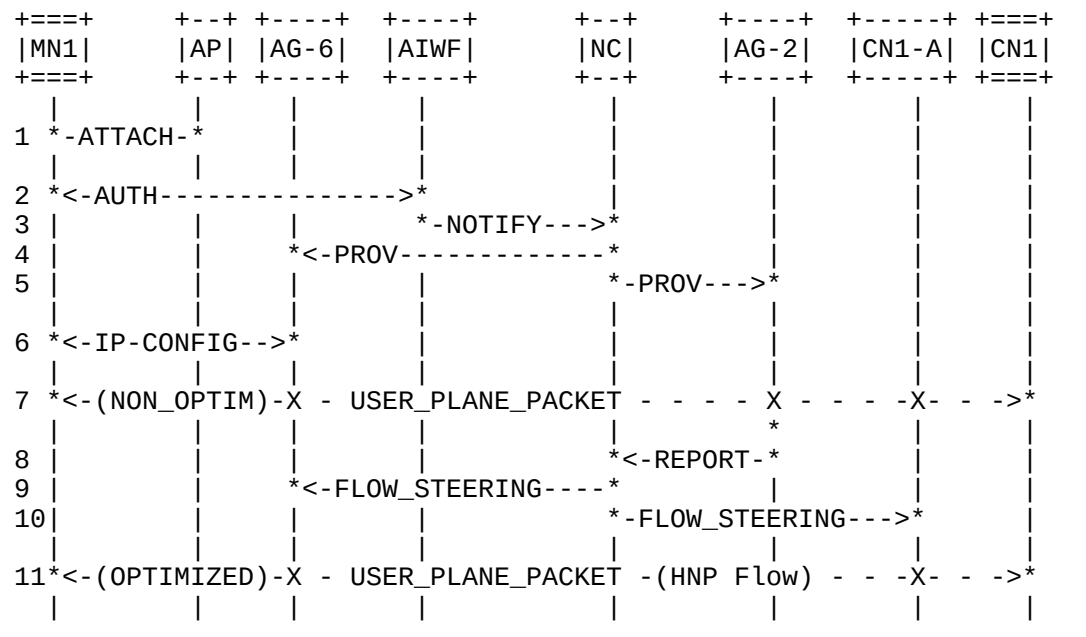


Figure 8: Mobile Node’s Roaming within the MFA Domain

- 1-ATTACH: The mobile node with NAI (MN1@ietf.org) roams in the network from AG-2 to AG-6.

- 2-AUTH: The mobile node completes the handover to the new access network using access network specific security mechanisms. The Auth-IWK function updates the mobile node’s location in the node-location database. The updated entry in the node location database will include the mobile node’s NAI, attach type as HANDOVER, and the current access-network location as AG-6.

- 3-NOTIFY: The Auth-IWK function function delivers the handover event to the MFA node controller. The information elements that are delivered include the mobile node identifier (MN-1@ietf.org), type of attach as HANDOVER, and the identity of the access gateway as AG-6.

- 4-IP_PROV: The NC provisions AG-6 for hosting the MN’s home-network prefix and local network prefix. The home network prefix, H1::/64 is from the previous anchor, AG-2 and is not topologically anchored on AG-6. However, for supporting mobility the prefix is hosted on the access link while the mobile node is attached to that access network and till there are active flows. The NC also

provisions AG-6 for hosting a new local network prefix, L2::/64. This prefix, L2::/64 is from a larger aggregate block that is topologically anchored on AG-6. The AG-6 will include meta-data in the IPv6 RA messages for indicating the properties of the prefixes; H1::/64 as the prefix with mobility support and L2::/64 as the prefix with no mobility support. The NC also provisions a traffic steering rule to steer all uplink IP traffic with source address H1::/64 through the previous anchor AG-2.

- o 5-IP_PROV: The NC provisions AG-2 to steer all IP traffic to destination addresses matching the prefix, H1::/64 to AG-6, and it also provisions a rule to report flow meta-data of those flows taking the non-optimal traffic path through AG-2. This essentially allows the NC to learn about any mobile node's IP flows still going through AG-2, so it can stitch the optimized path for those flows and remove AG-2 from the path for those flows.

- o 6-IP_CONFIG: The prefix H1::/64, obtained at the new location, will continue to be available on the new access link. The new local network prefix L2::/64 will also be available on the new access link and will be marked as a prefix with no mobility property. The mobile node may generate one, or more IPv6 addresses using the prefix L2::/64. The prefix L1::/64 is no longer hosted on the new link and the mobile node will remove it from interface configuration.

- o 7-USER_PLANE_PACKET: Any uplink IP link from CN1 will come to AG-2, as its the topological anchor for that address/prefix and AG-2 will steer the traffic directly to AG-6. On detecting an IP flow with the IP address belonging to prefix H1::/64, AG-2 will report the CN1-MN1 flow meta-data to NC.

- o 8-Report: The NC on receiving this event will lookup the CN anchor for the flow in its node location database. If the CN is another MN within the MFA domain, its current anchor information is retrieved from the node location database. However, if the CN is a node outside the MFA domain, the anchor for this node can be any transit router in the MFA domain which is always in path for that destination. The CN-anchor determination for nodes outside the MFA domain will be based on the network topology database.

- o 9-FLOW_STEERING: The NC inserts a IP traffic steering rule on AG-6 to steer the MN1-CN1's IP flows using H1::/64 directly to CN1's anchor which is CN1-A, and bypassing AG-2.

- o 10-FLOW_STEERING: The NC inserts a IP traffic steering rule on CN1-A to steer the MN1-CN1 IP flows using H1::/64 directly to

MN1's current anchor which is AG-6, and bypassing AG-2.

- 11-USER_PLANE_PACKET: The MN1-CN1's IP flows using H1::/64 will be steered directly from CN1-A to AG-6; AG-2 will not be in the path.

4.3. Traffic Steering State Removal

The mobile node's IP flows that were established at the previous location are no longer active. The steering state that was introduced at AG-6 and CN1-A will removed on detecting the inactive flows. The network may also optionally choose to withdraw the prefix H1::/64 and may assign a new HNP prefix which are topologically anchored in the new location.

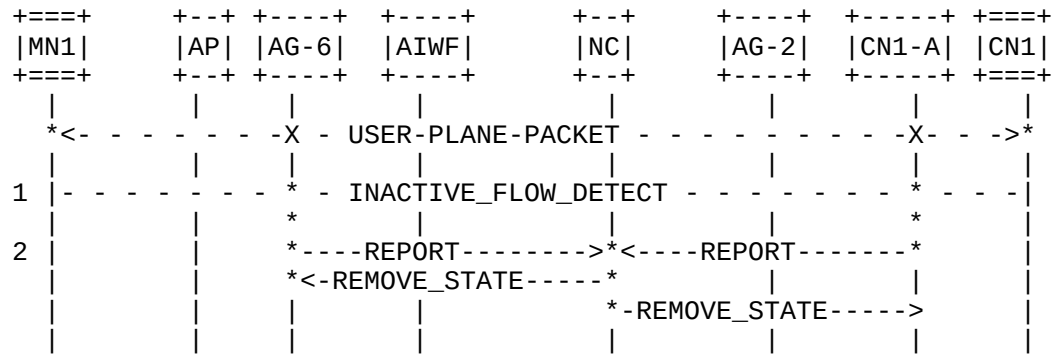


Figure 9: State Removal

- 1-IN_ACTIVE_FLOW_DETECT: At some point the MN1-CN1 flow using the prefix H1::/64 is no longer active.

- 2-REPORT: Both AG-6 and AG-2 will detect the inactive flows and may report this event to the NC. The steering state associated with MN1-CN1 flow using the prefix H1::/64 may be removed prior to reporting to the NC. Optionally, the NC on receiving the INACTIVE_FLOW_DETECT event may provision AG-6 and CN1-A to remove the steering state.

- 4-REMOVE_STATE:

4.4. Mobile Node's new IP flows

The mobile node's IP flows that were established at the previous location are no longer active and any created steering state was removed. The network may optionally choose to withdraw the prefix H1::/64 and may assign a new HNP prefix which is topologically anchored in the new location. All new IP flows will use the new prefix and the flows will take optimal routing path.

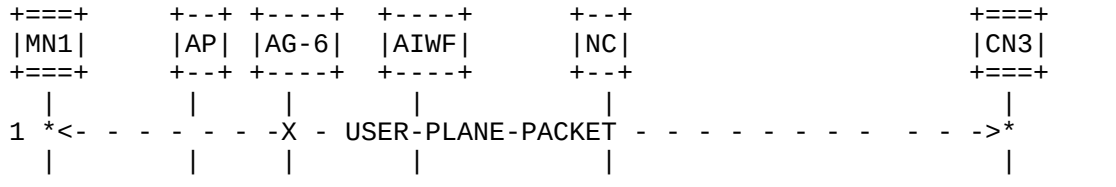


Figure 10: New Flows

- 1-USER_PLANE_PACKET: The mobile node's has established some IP flows using the IP address from the new HNP and LNP assigned at the new location. These IP flows will take optimal routing path and there is no need for any steering state, or the use of tunnels in the network for the mobile node's traffic.

5. MFA in 5G System Architecture

3GPP is specifying the 5G System Architecture, which follows a split between control- and data plane. Key control plane functions, which have interfaces to the data plane, are the Access Network and Mobility Management Function (AMF), and the Session Management Function (SMF). AMF and SMF cooperate to set up data plane nodes in the (radio) access network ((R)AN) and the core network, which comprises one or multiple User Plane Functions (UPF). As soon as a mobile node (UE) attaches to the network, as Packet Data Unit (PDU) Session is established and the SMF in the control plane selects one UPF as PDU Session Anchor, which serves also as IP address anchor. The SMF may select one more UPF on the path in between the PDU Session Anchor and the (R)AN, which enables routing traffic in between the UE and a local packet data network (PDN) with a correspondent node or service without the need to traverse the PDU Session Anchor.

In the view of MFA, each UPF can represent a locator for the UE's downlink traffic on the N9 as well as on the N6 reference point in

the 5G System Architecture. Since the SMF is in charge of UPF selection and configuration, the MFA-NC can leverage the SMF to retrieve node location information per this specification's procedure to access the NLDB from the MFA-NC. For MFA node selection and traffic steering, the MFA-NC may need more information about the data plane in terms of the transport network nodes and topology. Details about the NTDB are left out of this version of the document, but a realization may exploit available Topology information per [I-D.ietf-dmm-fpc-cpdp].

In the figure below, a UE's UPFs can function as MFA nodes, either as MFA-MNA or as MFA-CNA in case of mobile to mobile communication. Other transport network nodes, which may function as MFA-CNA for the UE's communication with a (non-mobile) correspondent node or service, are not explicitly depicted in the below figure. The MFA function can be tightly coupled with a UFP (co-located) or loosely coupled (separated). The MFA-NC utilized the FPC models and operation to enforce traffic steering policies in the MFA nodes. In case of loose coupling, the SMF utilizes the N4 protocol per the 3GPP standard to configure the selected UPF, whereas the MFA-NC uses FPC to enforce policies in the associated (loosely coupled) MFA node. In case of tight coupling, the MFA-NC may be co-located with the SMF and a single reference point and associated protocol may be used in between the SMF/MFA-NC and a UPF/MFA node.

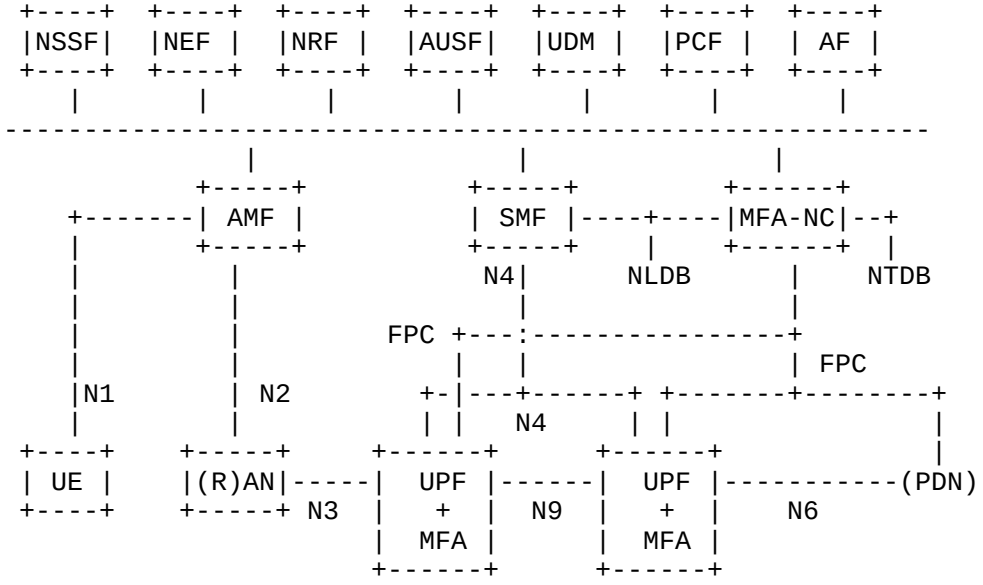


Figure 11: New Flows

6. IANA Considerations

TBD

7. Security Considerations

This specification allows a mobility node controller to provision IP traffic steering policies on the user plane nodes. It essentially leverages the FPC interface [I-D.ietf-dmm-fpc-cpdp] for interfacing with the user-plane anchor nodes. The security considerations specified in the FPC specification are sufficient for securing the messages carried on this interface.

The traffic steering rules that are provisioned on the MFA nodes by the MFA node controller are the standard policy rules that the FPC interface defines and does not require any new security considerations.

8. Acknowledgements

TBD

9. References

9.1. Normative References

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

9.2. Informative References

[I-D.filsfils-spring-srv6-network-programming] Filsfils, C., Leddy, J., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R., Matsushima, S., Lebrun, D., Decraene, B., Peirens, B., Salsano, S., Naik, G., Elmalky, H., Jonnalagadda, P., Sharif, M., Ayyangar, A., Mynam, S., Henderickx, W., Bashandy, A., Raza, K., Dukes, D., Clad, F., and P. Camarillo, "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming-03 (work in

progress), December 2017.

[I-D.ietf-dmm-fpc-cpdp]
Matsushima, S., Bertz, L., Liebsch, M., Gundavelli, S., Moses, D., and C. Perkins, "Protocol for Forwarding Policy Configuration (FPC) in DMM", draft-ietf-dmm-fpc-cpdp-09 (work in progress), October 2017.

[I-D.ietf-dmm-ondemand-mobility]
Yegin, A., Moses, D., Kweon, K., Lee, J., Park, J., and S. Jeon, "On Demand Mobility Management", draft-ietf-dmm-ondemand-mobility-13 (work in progress), January 2018.

[I-D.ietf-dmm-srv6-mobile-uplane]
Matsushima, S., Filsfils, C., Kohno, M., daniel.voyer@bell.ca, d., and C. Perkins, "Segment Routing IPv6 for Mobile User-Plane", draft-ietf-dmm-srv6-mobile-uplane-00 (work in progress), November 2017.

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

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

Authors' Addresses

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

Email: sgundave@cisco.com

Marco Liebsch
NEC
Kurfuersten-Anlage 36
D-69115 Heidelberg,
Germany

Email: liebsch@neclab.eu

Satoru Matsushima
SoftBank
Tokyo,
Japan

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

Distributed Mobility Management [dmm]
Internet-Draft
Intended status: Standards Track
Expires: May 17, 2018

C. Perkins
Futurewei
V. Devarapalli
Vasona Networks

November 13, 2017

MN Identifier Types for RFC 4283 Mobile Node Identifier Option
draft-ietf-dmm-4283mnids-06.txt

Abstract

Additional Identifier Type Numbers are defined for use with the Mobile Node Identifier Option for MIPv6 (RFC 4283).

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 17, 2018.

Copyright Notice

Table of Contents

1. Introduction

The Mobile Node Identifier Option for MIPv6 [RFC4283] has proved to be a popular design tool for providing identifiers for mobile nodes during authentication procedures with AAA protocols such as Diameter [RFC3588]. To date, only a single type of identifier has been specified, namely the MN NAI. Other types of identifiers are in common use, and even referenced in RFC 4283. In this document, we propose adding some basic types that are defined in various telecommunications standards, including types for IMSI [ThreeGPP-IDS], P-TMSI [ThreeGPP-IDS], IMEI [ThreeGPP-IDS], and GUTI [ThreeGPP-IDS]. In addition, we specify the IPv6 address itself and IEEE MAC-layer addresses as mobile node identifiers. Defining identifiers that are tied to the physical elements of the device (RFID, MAC address etc.) help in deployment of Mobile IP because in

many cases such identifiers are the most natural means for uniquely identifying the device, and will avoid additional look-up steps that might be needed if other identifiers were used.

2. Terminology

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 [RFC2119].

3. New Mobile Node Identifier Types

The following types of identifiers are commonly used to identify mobile nodes. For each type, references are provided with full details on the format of the type of identifer.

The Tag Data standard promoted by Electronic Product Code(TM) (abbreviated EPC) [EPC-Tag-Data] supports several encoding systems or schemes, which are commonly used in RFID (radio-frequency identification) applications, including

- RFID-GID (Global Identifier),
- RFID-SGTIN (Serialized Global Trade Item Number),
- RFID-SSCC (Serial Shipping Container),
- RFID-SGLN (Global Location Number),
- RFID-GRAI (Global Returnable Asset Identifier),
- RFID-DOD (Department of Defense ID), and
- RFID-GIAI (Global Individual Asset Identifier).

For each RFID scheme except GID, there are three representations:

- a 64-bit binary representation (for example, SGLN-64) (except for GID)
- a 96-bit binary representation (SGLN-96)
- a representation as a URI

The URI representation for the RFID is actually a URN. The EPC document has the following language:

> All categories of URIs are represented as Uniform Reference Names (URNs) as defined by [RFC2141], where the URN Namespace is epc.

The following list includes the above RFID types as well as various other common identifiers.

Mobile Node Identifier Description

Identifier Type	Description	Reference
IPv6 Address		[RFC4291]
IMSI	International Mobile Subscriber Identity	[ThreeGPP-IDS]
P-TMSI	Packet-Temporary Mobile Subscriber Identity	[ThreeGPP-IDS]
GUTI	Globally Unique Temporary ID	[ThreeGPP-IDS]
EUI-48 address	48-bit Extended Unique Identifier	[IEEE802]
EUI-64 address	64-bit Extended Unique Identifier-64 bit	[IEEE802]
DUID	DHCPv6 Unique Identifier	[RFC3315]
RFID-SGTIN-64	64-bit Serialized Global Trade Item Number	[EPC-Tag-Data]
RFID-SSCC-64	64-bit Serial Shipping Container	[EPC-Tag-Data]
RFID-SGLN-64	64-bit Serialized Global Location Number	[EPC-Tag-Data]
RFID-GRAI-64	64-bit Global Returnable Asset Identifier	[EPC-Tag-Data]
RFID-DOD-64	64-bit Department of Defense ID	[RFID-DoD-spec]
RFID-GIAI-64	64-bit Global Individual Asset Identifier	[EPC-Tag-Data]
RFID-GID-96	96-bit Global Identifier	[EPC-Tag-Data]
RFID-SGTIN-96	96-bit Serialized Global Trade Item Number	[EPC-Tag-Data]
RFID-SSCC-96	96-bit Serial Shipping Container	[EPC-Tag-Data]
RFID-SGLN-96	96-bit Serialized Global Location Number	[EPC-Tag-Data]
RFID-GRAI-96	96-bit Global Returnable Asset Identifier	[EPC-Tag-Data]
RFID-DOD-96	96-bit Department of Defense ID	[RFID-DoD-spec]
RFID-GIAI-96	96-bit Global Individual Asset Identifier	[EPC-Tag-Data]
RFID-GID-URI	Global Identifier represented as URI	[EPC-Tag-Data]
RFID-SGTIN-URI	Serialized Global Trade Item Number represented as URI	[EPC-Tag-Data]
RFID-SSCC-URI	Serial Shipping Container represented as URI	[EPC-Tag-Data]
RFID-SGLN-URI	Global Location Number represented as URI	[EPC-Tag-Data]

RFID-GRAI-URI	Global Returnable Asset Identifier represented as URI	[EPC-Tag-Data]
RFID-DOD-URI	Department of Defense ID represented as URI	[RFID-DoD-spec]
RFID-GIAI-URI	Global Individual Asset Identifier represented as URI	[EPC-Tag-Data]

Table 1

4. Descriptions of MNID types

In this section descriptions for the various MNID types are provided.

4.1. Description of the IPv6 address type

The IPv6 address [RFC4291] is encoded as a 16 octet string containing a full IPv6 address which has been assigned to the mobile node. The IPv6 address MUST be a unicast routable IPv6 address. Multicast addresses, link-local addresses, and the unspecified IPv6 address MUST NOT be used. IPv6 Unique Local Addresses (ULAs) MAY be used, as long as any security operations making use of the ULA also take into account the domain in which the ULA is guaranteed to be unique.

4.2. Description of the IMSI MNID type

The International Mobile Subscriber Identity (IMSI) [ThreeGPP-IDS] is at most 15 decimal digits (i.e., digits from 0 through 9). The IMSI MUST be encoded as a string of octets in network order (i.e., high-to-low for all digits), where each digit occupies 4 bits. If needed for full octet size, the last digit MUST be padded with 0xf. For example an example IMSI 123456123456789 would be encoded as follows:

0x12, 0x34, 0x56, 0x12, 0x34, 0x56, 0x78, 0x9f

4.3. Description of the EUI-48 address type

The IEEE EUI-48 address [IEEE802-eui48] is encoded as 6 octets containing the IEEE EUI-48 address.

4.4. Description of the EUI-64 address type

The IEEE EUI-64 address [IEEE802-eui64] is encoded as 8 octets containing the full IEEE EUI-64 address.

4.5. Description of the DUID type

The DUID is the DHCPv6 Unique Identifier (DUID) [RFC3315]. There are various types of DUID, which are distinguished by an initial two-octet type field. Clients and servers MUST treat DUIDs as opaque values and MUST only compare DUIDs for equality.

4.6. Description of the RFID types

The General Identifier (GID) that is used with RFID is composed of three fields - the General Manager Number, Object Class and Serial Number. The General Manager Number identifies an organizational entity that is responsible for maintaining the numbers in subsequent fields. GID encodings include a fourth field, the header, to guarantee uniqueness in the namespace defined by EPC.

Some of the RFID types depend on the Global Trade Item Number (GTIN) code defined in the General EAN.UCC Specifications [EANUCCGS]. A GTIN identifies a particular class of object, such as a particular kind of product or SKU.

The EPC encoding scheme for SGTIN permits the direct embedding of EAN.UCC System standard GTIN and Serial Number codes on EPC tags. In all cases, the check digit is not encoded. Two encoding schemes are specified, SGTIN-64 (64 bits) and SGTIN-96 (96 bits).

The Serial Shipping Container Code (SSCC) is defined by the EAN.UCC Specifications. Unlike the GTIN, the SSCC is already intended for assignment to individual objects and therefore does not require additional fields to serve as an EPC pure identity. Two encoding schemes are specified, SSCC-64 (64 bits) and SSCC-96 (96 bits).

The Global Location Number (GLN) is defined by the EAN.UCC Specifications. A GLN can represent either a discrete, unique physical location such as a warehouse slot, or an aggregate physical location such as an entire warehouse. In addition, a GLN can represent a logical entity that performs a business function such as placing an order. The Serialized Global Location Number (SGLN) includes the Company Prefix, Location Reference, and Serial Number.

The Global Returnable Asset Identifier (GRAI) is defined by the General EAN.UCC Specifications. Unlike the GTIN, the GRAI is already intended for assignment to individual objects and therefore does not require any additional fields to serve as an EPC pure identity. The GRAI includes the Company Prefix, Asset Type, and Serial Number.

The Global Individual Asset Identifier (GIAI) is defined by the General EAN.UCC Specifications. Unlike the GTIN, the GIAI is already

intended for assignment to individual objects and therefore does not require any additional fields to serve as an EPC pure identity. The GRAI includes the Company Prefix, and Individual Asset Reference.

The DoD Construct identifier is defined by the United States Department of Defense (DoD). This tag data construct may be used to encode tags for shipping goods to the DoD by a supplier who has already been assigned a CAGE (Commercial and Government Entity) code.

4.6.1. Description of the RFID-SGTIN-64 type

The RFID-SGTIN-64 is encoded as specified in [EPC-Tag-Data]. The SGTIN-64 includes five fields: Header, Filter Value (additional data that is used for fast filtering and pre-selection), Company Prefix Index, Item Reference, and Serial Number. Only a limited number of Company Prefixes can be represented in the 64-bit tag.

4.6.2. Description of the RFID-SGTIN-96 type

The RFID-SGTIN-96 is encoded as specified in [EPC-Tag-Data]. The SGTIN-96 includes six fields: Header, Filter Value, Partition (an indication of where the subsequent Company Prefix and Item Reference numbers are divided), Company Prefix Index, Item Reference, and Serial Number.

4.6.3. Description of the RFID-SSCC-64 type

The RFID-SSCC-64 is encoded as specified in [EPC-Tag-Data]. The SSCC-64 includes four fields: Header, Filter Value, Company Prefix Index, and Serial Reference. Only a limited number of Company Prefixes can be represented in the 64-bit tag.

4.6.4. Description of the RFID-SSCC-96 type

The RFID-SSCC-96 is encoded as specified in [EPC-Tag-Data]. The SSCC-96 includes six fields: Header, Filter Value, Partition, Company Prefix, and Serial Reference, as well as 24 bits that remain Unallocated and must be zero.

4.6.5. Description of the RFID-SGLN-64 type

The RFID-SGLN-64 type is encoded as specified in [EPC-Tag-Data]. The SGLN-64 includes five fields: Header, Filter Value, Company Prefix Index, Location Reference, and Serial Number.

4.6.6. Description of the RFID-SGLN-96 type

The RFID-SGLN-96 type is encoded as specified in [EPC-Tag-Data]. The SGLN-96 includes six fields: Header, Filter Value, Partition, Company Prefix, Location Reference, and Serial Number.

4.6.7. Description of the RFID-GRAI-64 type

The RFID-GRAI-64 type is encoded as specified in [EPC-Tag-Data]. The GRAI-64 includes five fields: Header, Filter Value, Company Prefix Index, Asset Type, and Serial Number.

4.6.8. Description of the RFID-GRAI-96 type

The RFID-GRAI-96 type is encoded as specified in [EPC-Tag-Data]. The GRAI-96 includes six fields: Header, Filter Value, Partition, Company Prefix, Asset Type, and Serial Number.

4.6.9. Description of the RFID-GIAI-64 type

The RFID-GIAI-64 type is encoded as specified in [EPC-Tag-Data]. The GIAI-64 includes four fields: Header, Filter Value, Company Prefix Index, and Individual Asset Reference.

4.6.10. Description of the RFID-GIAI-96 type

The RFID-GIAI-96 type is encoded as specified in [EPC-Tag-Data]. The GIAI-96 includes five fields: Header, Filter Value, Partition, Company Prefix, and Individual Asset Reference.

4.6.11. Description of the RFID-DoD-64 type

The RFID-DoD-64 type is encoded as specified in [RFID-DoD-spec]. The DoD-64 type includes four fields: Header, Filter Value, Government Managed Identifier, and Serial Number.

4.6.12. Description of the RFID-DoD-96 type

The RFID-DoD-96 type is encoded as specified in [RFID-DoD-spec]. The DoD-96 type includes four fields: Header, Filter Value, Government Managed Identifier, and Serial Number.

4.6.13. Description of the RFID URI types

In some cases, it is desirable to encode in URI form a specific encoding of an RFID tag. For example, an application may prefer a URI representation for report preparation. Applications that wish to

manipulate any additional data fields on tags may need some representation other than the pure identity forms.

For this purpose, the fields as represented the previous sections are associated with specified fields in the various URI types. For instance, the URI may have fields such as CompanyPrefix, ItemReference, or SerialNumber. For details and encoding specifics, consult [EPC-Tag-Data].

5. Security Considerations

This document does not introduce any security mechanisms, and does not have any impact on existing security mechanisms.

Mobile Node Identifiers such as those described in this document are considered to be private information. If used in the MNID extension as defined in [RFC4283], the packet including the MNID extension MUST be encrypted so that no personal information or trackable identifiers is inadvertently disclosed to passive observers. Operators can potentially apply IPsec Encapsulating Security Payload (ESP) [RFC4303], in transport mode, with confidentiality and integrity protection for protecting the identity and location information in Mobile IPv6 signaling messages.

Some MNIDs contain sensitive identifiers which, as used in protocols specified by other SDOs, are only used for signaling during initial network entry. In such protocols, subsequent exchanges then rely on a temporary identifier allocated during the initial network entry. Managing the association between long-lived and temporary identifiers is outside the scope of this document.

6. IANA Considerations

The new mobile node identifier types defined in the document should be assigned values from the "Mobile Node Identifier Option Subtypes" registry. The following values should be assigned.

New Mobile Node Identifier Types

Identifier Type	Identifier Type Number
IPv6 Address	2
IMSI	3
P-TMSI	4
EUI-48 address	5
EUI-64 address	6
GUTI	7
DUID-LLT	8
DUID-EN	9
DUID-LL	10
DUID-UUID	11
	12-15 reserved
	16 reserved
RFID-SGTIN-64	17
RFID-SSCC-64	18
RFID-SGLN-64	19
RFID-GRAI-64	20
RFID-DOD-64	21
RFID-GIAI-64	22
	23 reserved
RFID-GID-96	24
RFID-SGTIN-96	25
RFID-SSCC-96	26
RFID-SGLN-96	27
RFID-GRAI-96	28
RFID-DOD-96	29
RFID-GIAI-96	30
	31 reserved
RFID-GID-URI	32
RFID-SGTIN-URI	33
RFID-SSCC-URI	34
RFID-SGLN-URI	35
RFID-GRAI-URI	36
RFID-DOD-URI	37
RFID-GIAI-URI	38
	39-255 unassigned

Table 2

See Section 4 for additional information about the identifier types. Future new assignments are to be made only after Expert Review [RFC8126]. The expert must ascertain that the identifier type allows unique identification of the mobile device; since all MNIDs require

encryption there is no additional privacy exposure attendent to the use of new types.

7. Acknowledgements

The authors wish to acknowledge Hakima Chaouchi, Tatuya Jinmei, Jouni Korhonen, Sri Gundavelli, Suresh Krishnan, Dapeng Liu, Dale Worley, Joseph Salowey, Linda Dunbar, and Mirja Kuehlewind for their helpful comments.

8. References

8.1. Normative References

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

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

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC4283] Patel, A., Leung, K., Khalil, M., Akhtar, H., and K. Chowdhury, "Mobile Node Identifier Option for Mobile IPv6 (MIPv6)", RFC 4283, DOI 10.17487/RFC4283, November 2005, <https://www.rfc-editor.org/info/rfc4283>.

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

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <https://www.rfc-editor.org/info/rfc4303>.

[RFC6355] Narten, T. and J. Johnson, "Definition of the UUID-Based DHCPv6 Unique Identifier (DUID-UUID)", RFC 6355, DOI 10.17487/RFC6355, August 2011, <https://www.rfc-editor.org/info/rfc6355>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

8.2. Informative References

[EANUCCGS]
EAN International and the Uniform Code Council, "General EAN.UCC Specifications Version 5.0", Jan 2004.

[EPC-Tag-Data]
EPCglobal Inc., "EPC(TM) Generation 1 Tag Data Standards Version 1.1 Rev.1.27 http://www.gs1.org/gsmp/kc/epcglobal/tds/ tds_1_1_rev_1_27-standard-20050510.pdf", January 2005.

[IEEE802] IEEE, "IEEE Std 802: IEEE Standards for Local and Metropolitan Networks: Overview and Architecture", 2001.

[IEEE802-eui48]
IEEE, "Guidelines for 48-Bit Global Identifier (EUI-48) https://standards.ieee.org/develop/regauth/tut/eui48.pdf", 2001.

[IEEE802-eui64]
IEEE, "Guidelines for 64-Bit Global Identifier (EUI-64) https://standards.ieee.org/develop/regauth/tut/eui.pdf64", 2001.

[RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, DOI 10.17487/RFC3588, September 2003, <https://www.rfc-editor.org/info/rfc3588>.

[RFID-DoD-spec]
Department of Defense, "United States Department of Defense Suppliers Passive RFID Information Guide (Version 15.0)", January 2010.

[ThreeGPP-IDS]
3rd Generation Partnership Project, "3GPP Technical Specification 23.003 V8.4.0: Technical Specification Group Core Network and Terminals; Numbering, addressing and identification (Release 8)", March 2009.

Authors' Addresses

Charles E. Perkins
Futurewei Inc.
2330 Central Expressway
Santa Clara, CA 95050
USA

Phone: +1-408-330-4586
Email: charliep@computer.org

Vijay Devarapalli
Vasona Networks
2900 Lakeside Drive, Suite 180
Santa Clara, CA 95054
USA

Email: dvijay@gmail.com

DMM WG
Internet-Draft
Intended status: Informational
Expires: May 16, 2018

S. Gundavelli
Cisco
S. Jeon
Sungkyunkwan University

November 12, 2017

DMM Deployment Models and Architectural Considerations
draft-ietf-dmm-deployment-models-03.txt

Abstract

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

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 16, 2018.

Copyright Notice

Table of Contents

1. Overview

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

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

2. Conventions and Terminology

2.1. Conventions

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

2.2. Terminology

All the mobility related terms are to interpreted as defined in [RFC6275], [RFC5213], [RFC5844], [RFC7333], [RFC7665], [RFC7429], [I-D.ietf-sfc-nsh] and [I-D.ietf-dmm-fpc-cpdp]. Additionally, this document uses the following terms:

Home Control-Plane Anchor (H-CPA)

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

Home Data Plane Anchor (Home-DPA)

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

Access Control Plane Node (Access-CPN)

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

Access Data Plane Node (Access-DPN)

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

Routing Controller (RC)

The Routing Controller is a centralized control entity, which is able to instruct the forwarding behavior for mobility management in Home-DPA and Access-DPN.

Mobility Controller (MC)

The Mobility Controller is a function entity, which is able to manage the orchestration of Home-CPA and Access-CPN functions.

3. DMM Architectural Overview

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

1. Separation of control and data Plane
2. Aggregation of control plane for elastic scaling
3. Distribution of the data plane for efficient network usage
4. Elimination of mobility state from the data plane
5. Dynamic selection of control and data plane nodes
6. Enabling the mobile node with network properties
7. Relocation of anchor functions for efficient network usage

3.1. DMM Service Primitives

The functions in the DMM architecture support a set of service primitives. Each of these service primitives identifies a specific service capability with the exact service definition. The functions in the DMM architecture are required to support a specific set of service primitives that are mandatory for that service function. Not all service primitives are applicable to all DMM functions. The below table as shown in Fig. 1 identifies the service primitives that each of the DMM function SHOULD support. The marking "X" indicates the service primitive on that row needs to be supported by the identified DMM function on the corresponding column; for example, the IP address management must be supported by Home-CPA function. The NSH Classifier denotes the SFC entity that performs the classification of a service flow, defined in [RFC7665].

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

Figure 1: Mapping of DMM functions

3.2. DMM Functions and Interfaces

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

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

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

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

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

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

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

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

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

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

3.2.3. Access Control Plane Node (Access-CPN)

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

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

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

3.2.4. Access Data Plane Node (Access-DPN)

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

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

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

3.2.5. DMM Functions Mapping to Other Architectures

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

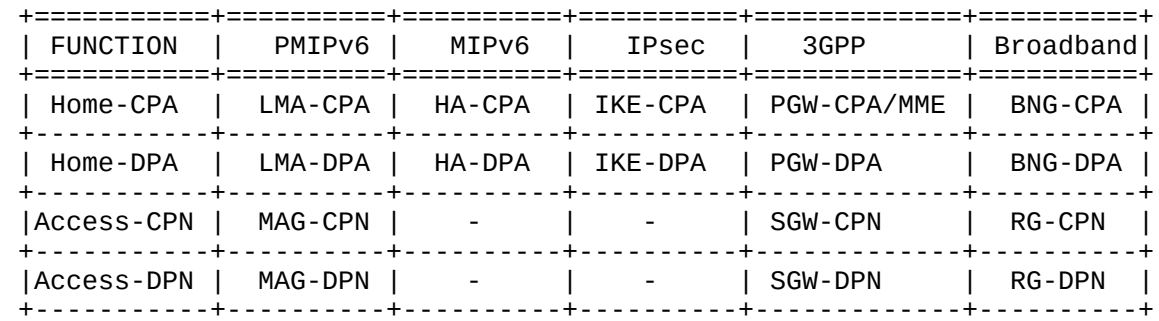


Figure 2: Mapping of DMM functions

4. Deployment Models

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

4.1. Model-1: Split Home Anchor Mode

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

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

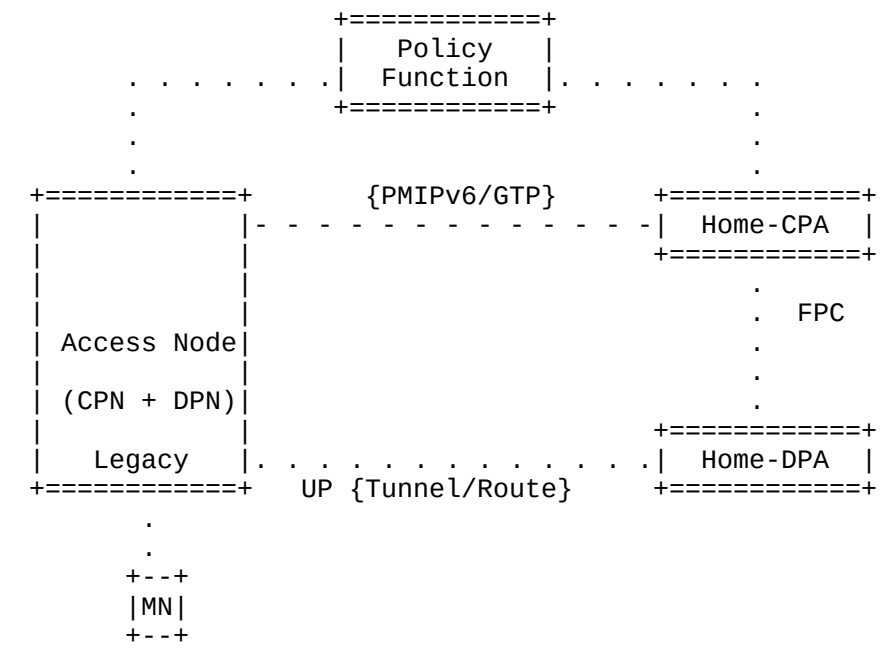


Figure 3: Split Home Anchor Mode

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

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

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

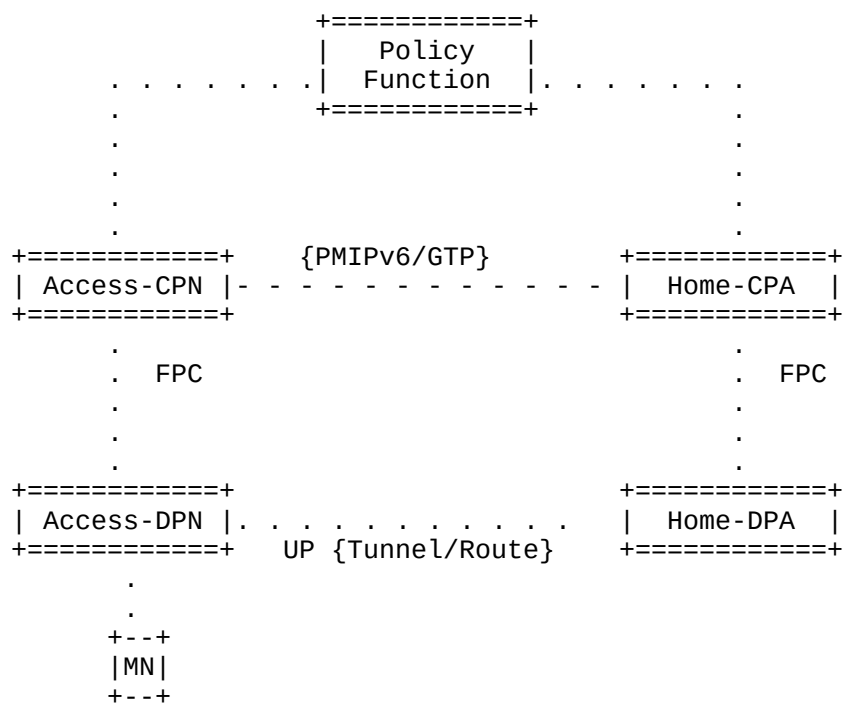


Figure 4: Seperated Control and User Plane Mode

4.3. Model-3: Centralized Control Plane Mode

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

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

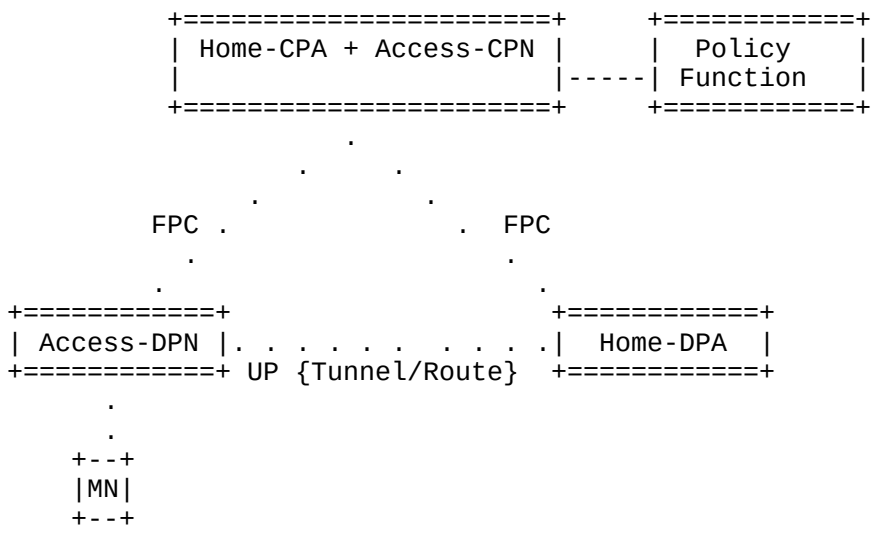


Figure 5: Centralized Control Plane Mode

4.4. Model-4: Data Plane Abstraction Mode

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

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

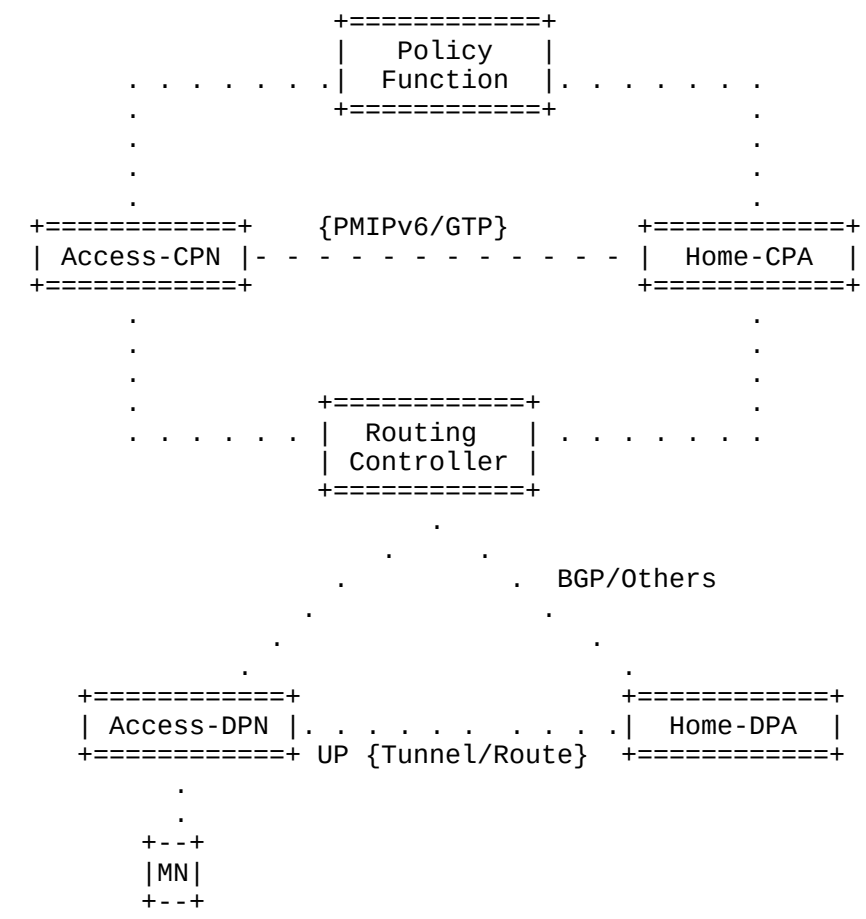


Figure 6: Data Plane Abstraction Mode

4.5. On-Demand Control Plane Orchestration Mode

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

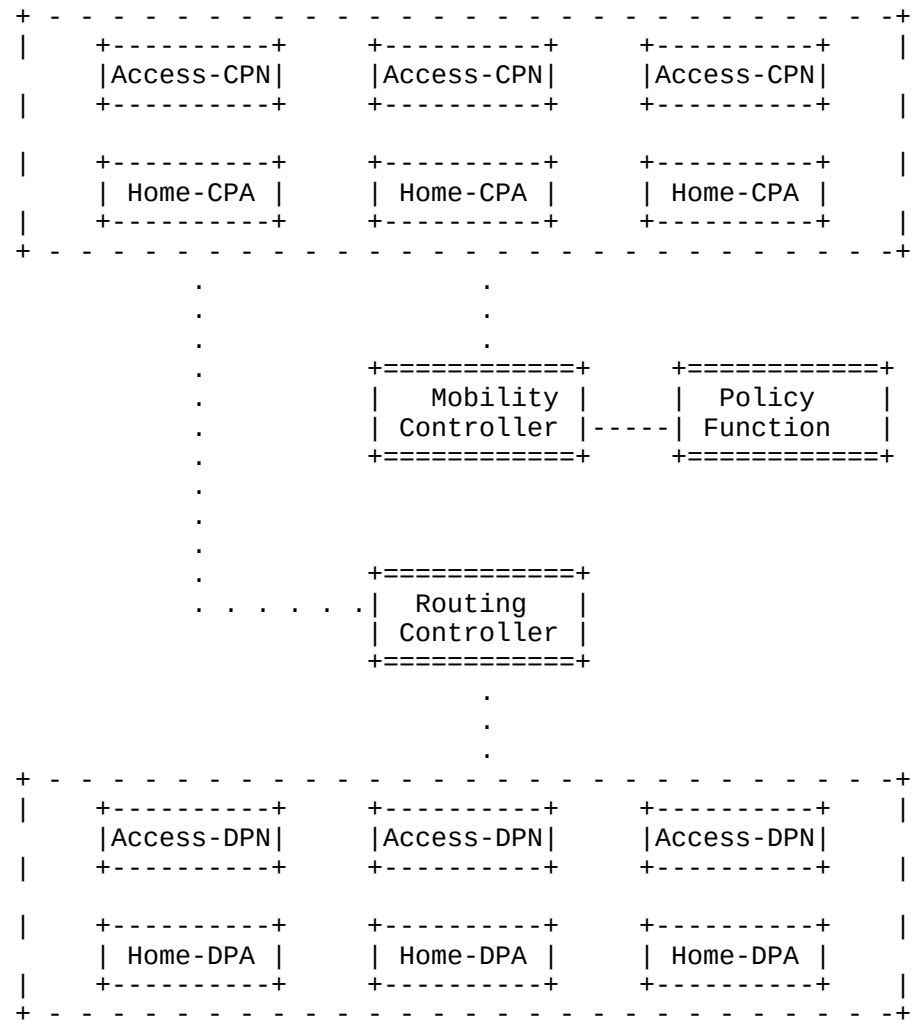


Figure 7: On-Demand CP Orchestration Mode

5. IANA Considerations

This document does not require any IANA actions.

6. Security Considerations

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

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

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

7. Work Team

This document reflects contributions from the following work team members:

Younghan Kim

younghak@ssu.ac.kr

Vic Liu

liuzhiheng@chinamobile.com

Danny S Moses

danny.moses@intel.com

Marco Liebsch

liebsch@neclab.eu

Carlos Jesus Bernardos Cano

cjbc@it.uc3m.es

8. Acknowledgements

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

9. References

9.1. Normative References

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

9.2. Informative References

[I-D.ietf-dmm-fpc-cpdp] Matsushima, S., Bertz, L., Liebsch, M., Gundavelli, S., Moses, D., and C. Perkins, "Protocol for Forwarding Policy Configuration (FPC) in DMM", draft-ietf-dmm-fpc-cpdp-09 (work in progress), October 2017.

[I-D.ietf-sfc-nsh] Quinn, P., Elzur, U., and C. Pignataro, "Network Service Header (NSH)", draft-ietf-sfc-nsh-28 (work in progress), November 2017.

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

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

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

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

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

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

Authors' Addresses

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

Email: sgundave@cisco.com

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

Email: seiljeon@skku.edu

DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 20, 2018

S. Matsushima
SoftBank
L. Bertz
Sprint
M. Liebsch
NEC
S. Gundavelli
Cisco
D. Moses
Intel Corporation
C. Perkins
Futurewei
June 18, 2018

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

Abstract

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

Status of This Memo

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.

Table of Contents

1. Introduction

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

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

A FPC agent represents the data-plane operation according to several basic information models. A FPC agent also provides access to Monitors, which produce reports when triggered by events or FPC Client requests regarding Mobility Contexts, DPNs or the Agent.

To manage mobility sessions, the FPC client assembles applicable sets of forwarding policies from the data model, and configures them on the appropriate FPC Agent. The Agent then renders those policies into specific configurations for each DPN at which mobile nodes are attached. The specific protocols and configurations to configure a DPN from a FPC Agent are outside the scope of this document.

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

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

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

2. Terminology

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

Attribute Expression: The definition of a template Property. This includes setting the type, current value, default value and if the attribute is static, i.e. can no longer be changed.

Domain:	One or more DPNs that form a logical partition of network resources (e.g., a data-plane network under common network administration). A FPC client (e.g., a mobility management system) may utilize a single or multiple domains.
DPN:	A data-plane node (DPN) is capable of performing data-plane features. For example, DPNs may be switches or routers, regardless of whether they are realized as hardware or purely in software.
FPC Client:	A FPC Client is integrated with a mobility management system or related application, enabling control over forwarding policy, mobility sessions and DPNs via a FPC Agent.
Mobility Context:	A Mobility Context contains the data-plane information necessary to efficiently send and receive traffic from a mobile node. This includes policies that are created or modified during the network's operation - in most cases, on a per-flow or per session basis. A Mobility-Context represents the mobility sessions (or flows) which are active on a mobile node. This includes associated runtime attributes, such as tunnel endpoints, tunnel identifiers, delegated prefix(es), routing information, etc. Mobility-Contexts are associated to specific DPNs. Some pre-defined Policies may apply during mobility signaling requests. The Mobility Context supplies information about the policy settings specific to a mobile node and its flows; this information is often quite dynamic.
Mobility Session:	Traffic to/from a mobile node that is expected to survive reconnection events.
Monitor:	A reporting mechanism for a list of events that trigger notification messages from a FPC Agent to a FPC Client.
Policy:	A Policy determines the mechanisms for managing specific traffic flows or packets. Policies specify QoS, rewriting rules for

	packet processing, etc. A Policy consists of one or more rules. Each rule is composed of a Descriptor and Actions. The Descriptor in a rule identifies packets (e.g., traffic flows), and the Actions apply treatments to packets that match the Descriptor in the rule. Policies can apply to Domains, DPNs, Mobile Nodes, Service-Groups, or particular Flows on a Mobile Node.
Property:	An attribute-value pair for an instance of a FPC entity.
Service-Group:	A set of DPN interfaces that support a specific data-plane purpose, e.g. inbound/ outbound, roaming, subnetwork with common specific configuration, etc.
Template:	A recipe for instantiating FPC entities. Template definitions are accessible (by name or by a key) in an indexed set. A Template is used to create specific instances (e.g., specific policies) by assigning appropriate values into the Template definition via Attribute Expression.
Template Configuration	The process by which a Template is referenced (by name or by key) and Attribute Expressions are created that change the value, default value or static nature of the Attribute, if permitted. If the Template is Extensible, new attributes MAY be added.
Tenant:	An operational entity that manages mobility management systems or applications which require data-plane functions. A Tenant defines a global namespace for all entities owned by the Tenant enabling its entities to be used by multiple FPC Clients across multiple FPC Agents.
Topology:	The DPNs and the links between them. For example, access nodes may be assigned to a Service-Group which peers to a Service-Group of anchor nodes.

3. FPC Design Objectives and Deployment

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

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

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

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

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

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

This document does not specify a protocol for the FPC interface; it is out of scope. However, an implementation must support the FPC transactions described in Section 5.

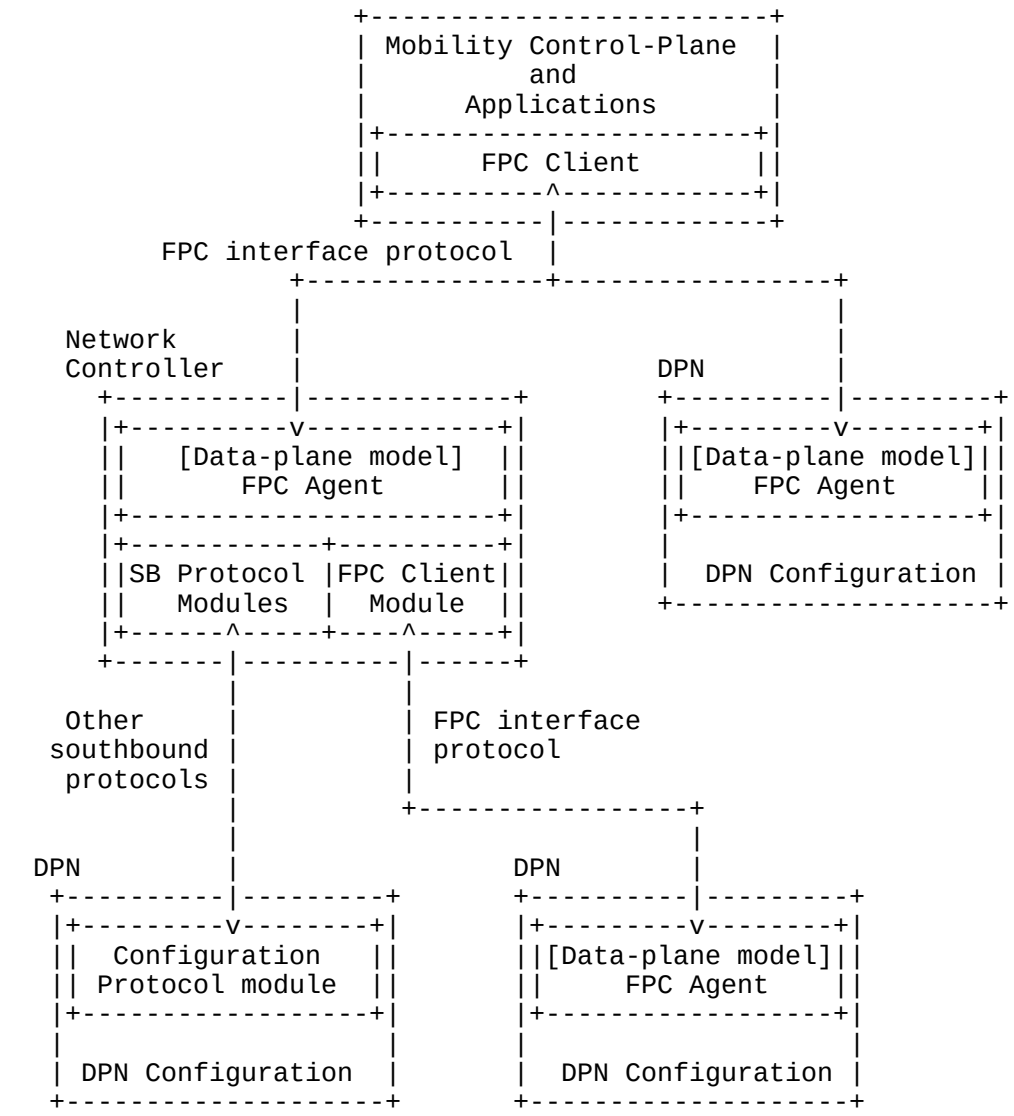


Figure 1: Reference Forwarding Policy Configuration (FPC) Architecture

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

Multi-tenancy permits the paritioning of data-plane entities as well as a common namespace requirement upon FPC Agents and Clients when they use the same Tenant for a common data-plane entity.

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

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

4. FPC Mobility Information Model

The FPC information model includes the following components:

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

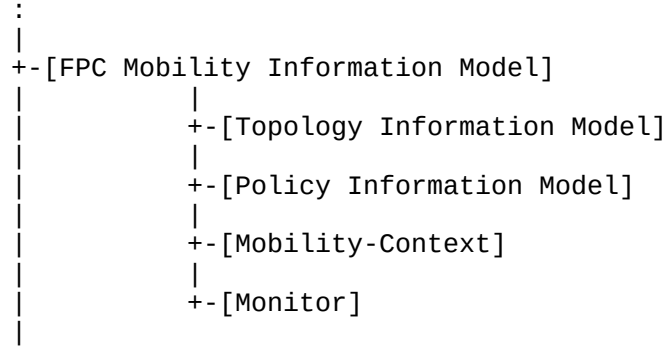


Figure 2: FPC Information Model structure

4.1. Model Notation and Conventions

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

Information model entities (e.g. DPNs, Rules, etc.) are defined in a hierarchical notation where all entities at the same hierarchical

level are located on the same left-justified vertical position sequentially. When entities are composed of sub-entities, the sub-entities appear shifted to the right, as shown in Figure 3.

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

Figure 3: Model Notation - An Example

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

List: A collection of entities (some could be duplicated)

Set: A nonempty collection of entities without duplications

Name: A human-readable string

Key: A unique value. We distinguish 3 types of keys:

U-Key: A key unique across all Tenants. U-Key spaces typically involve the use of registries or language specific mechanisms that guarantee universal uniqueness of values.

G-Key: A key unique within a Tenant

L-Key: A key unique within a local namespace. For example, there may exist interfaces with the same name, e.g. "if0", in two different DPNs but there can only be one "if0" within each DPN (i.e. its local Interface-Key L-Key space).

Each entity or attribute may be optional (O) or mandatory (M). Entities that are not marked as optional are mandatory.

```
The following example shows 3 entities:
    -- Entity1 is a globally unique key, and optionally can have
               an associated Name
    -- Entity2 is a list
    -- Entity3 is a set and is optional
               +
               |
               +-[entity1] <G-Key> (M), <Name> (O)
               +-[entity2] <List>
               +-[entity3] <Set> (O)
               |
               +
```

Figure 4

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

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

EntityA definition references an entityB model element.

EntityB model elements are indexed by entityB-Key.

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

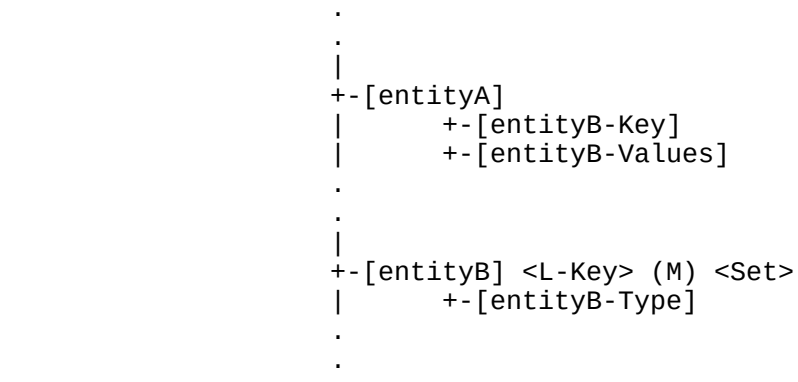


Figure 5: Indexed sets of entities

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

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

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

4.2. Templates and Attributes

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

```
+-[Template] <U-Key, Name> (M) <Set>
|      +-[Attributes] <Set> (M)
|      +-[Extensible ˜ FALSE]
|      +-[Entity-State ˜ Initial]
|      +-[Version]
```

Figure 6: Template entities

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

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

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

Version: Provides a version tag for the Template.

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

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

4.3. Attribute-Expressions

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

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

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

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

'[Att-Name]' Non-mandatory Attribute may be included but template does not provide any configured value.

'[Att-Name = Att-Value]' Non-mandatory Attribute may be included and has a statically configured value.

'[Att-Name ~ Att-Value]' Non-mandatory Attribute may be included and has a default value.

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

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

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

4.4. Attribute Value Types

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

4.5. Namespace and Format

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

When a Policy needs to be applied to Mobility-Contexts in all Tenants on an Agent, the Agent SHOULD define that policy to be visible by all Tenants. In this case, the Agent assigns a unique identifier in the

Agent namespace and copies the values to each Tenant. This effectively creates a U-Key although only a G-Key is required within the Tenant.

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

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

4.6. Configuring Attribute Values

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

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

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

Agent, on behalf of the FPC Client, applies or deactivates whatever policies and attribute values are appropriate for managing the flows of the mobile node. When a "Configured" policy is de-activated, Policy-Status is changed to be "Active". When an "Active" policy is activated, Policy-Status is changed to be "Configured".

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

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

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

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

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

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

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

Entity blocks describe the format of the policy configurations.

4.7. Entity Configuration Blocks

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

assign new values if the Attribute is not static

make attributes static if they were not

make an attribute mandatory

A Policy-Configuration MUST NOT define or refine an attribute twice. More generally, an Entity-Configuration can be defined for any configurable Indexed Set to be the combination of the Entity-Key along with a set of Attribute-Expressions that supply configuration information for the entity's Attributes. Figure 7 shows a schematic representation for such Entity Configuration Blocks.

```
[Entity Configuration Block]
|       +-[Entity-Key] (M)
|       +-[Attribute-Expression] <Set> (M)
```

Figure 7: Entity Configuration Block

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

Descriptor-Configuration

Action-Configuration

Rule-Configuration

Interface-Configuration

Service-Group-Configuration

Domain-Policy-Configuration

DPN-Policy-Configuration

Policy-Configuration

MN-Policy-Configuration

Service-Data-Flow-Policy-Configuration

4.8. Information Model Checkpoint

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

Checkpoints are defined for the following information model components:

Service-Group

DPN Information Model

Domain Information Model

Policy Information Model

4.9. Information Model Components

4.9.1. Topology Information Model

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

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

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

Figure 8: Topology Structure

4.9.2. Service-Group

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

```
|
+-[Service-Group] <G-Key>, <Name> (O) <Set>
|           +-[Extensible: FALSE]
|           +-[Role] <U-Key>
|           +-[Protocol] <Set>
|           +-[Feature] <Set> (O)
|           +-[Service-Group-Configuration] <Set> (O)
|           +-[DPN-Key] <Set>
|           |           +-[Referenced-Interface] <Set>
|           |           |       +-[Interface-Key] <L-Key>
|           |           |       +-[Peer-Service-Group-Key] <Set> (O)
```

Figure 9: Service Group

Each Service-Group element contains the following information:

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

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

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

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

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

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

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

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

Interface-Key: A key that is used together with the DPN-Key, to create a key that is refers to a specific DPN interface definition.

Peer-Service-Group-Key: Enables location of the peer Service-Group for this Interface.

4.9.3. Domain Information Model

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

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

Figure 10: Domain Information Model

Each Domain entry contains the following information:

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

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

4.9.4. DPN Information Model

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

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

Figure 11: DPN Information Model

Each DPN entry contains the following information:

DPN-Key: A unique Identifier of the DPN.

DPN-Name: A human-readable display string.

Domain-Key: A Key providing access to the Domain information about the Domain in which the DPN resides.

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

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

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

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

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

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

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

4.9.5. Policy Information Model

The Policy Information Model defines and identifies Rules for enforcement at DPNs. A Policy is basically a set of Rules that are to be applied to each incoming or outgoing packet at a DPN interface. Rules comprise Descriptors and a set of Actions. The Descriptors,

when evaluated, determine whether or not a set of Actions will be performed on the packet. The Policy structure is independent of a policy context.

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

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

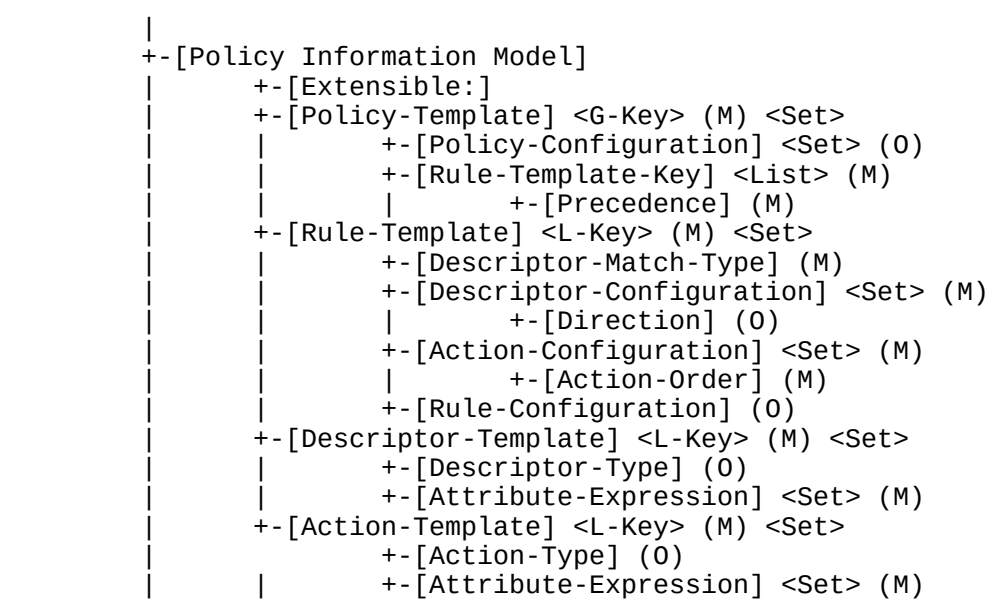


Figure 12: Policy Information Model

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.

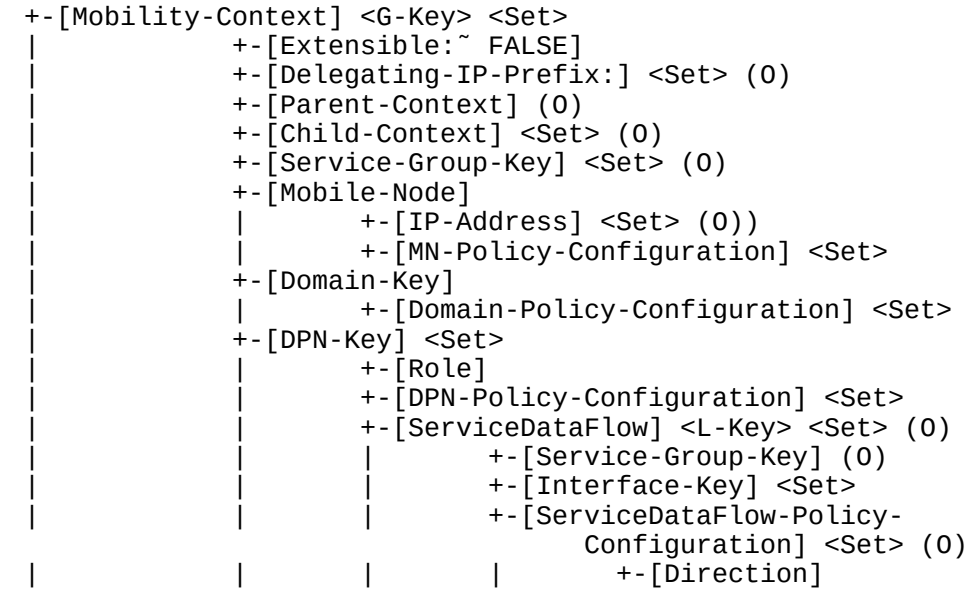


Figure 13: Mobility-Context Information Model

The Mobility-Context Substructure holds the following entries:

Mobility-Context-Key: Identifies a Mobility-Context

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

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

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

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

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

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

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

Domain-Key: Enables access to a Domain instance.

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

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

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

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

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

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

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

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

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

4.9.7. Monitor Information Model

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

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

```
                          |
                          +-[Monitor] <G-Key> <List>
                          |         +-[Extensible:]
                          |         +-[Target:]
                          |         +-[Deferrable]
                          |         +-[Configuration]
```

Figure 14: Monitor Substructure

Monitor-Key: Identifies the Monitor.

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

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

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

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

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

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

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

5. Protocol

5.1. Protocol Messages and Semantics

Four Client to Agent messages are supported.

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

Table 1: Client to Agent Messages

Each message contains a header with the following information:

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

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

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

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

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

Configure Result Notification from the Agent to the Client, see Section 5.1.1.4.2.

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

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

Error-Tag: An error tag.

Error-App-Tag: Application specific error tag.

Error-Message: A message describing the error.

Error-Info: Any data required for the response.

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

Figure 15: Error Information Model

Two Agent to Client notifications are supported.

Message	Description
Configure-Result-Notification	An asynchronous notification from Agent to Client based upon a previous Configure request.
Notify	An asynchronous notification from Agent to Client based upon a registered Monitor's configuration, a Monitor deregistration or Probe.

Table 2: Agent to Client Messages (notifications)

5.1.1. Configure Message

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

5.1.1.1. Edit Operation Types

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

Table 3: Configure Edit Operations

5.1.1.2. Edit Operation

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

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

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

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

Reference-Scope: If supported, specifies the Reference Scope (see Section 5.1.1.3)

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

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

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

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

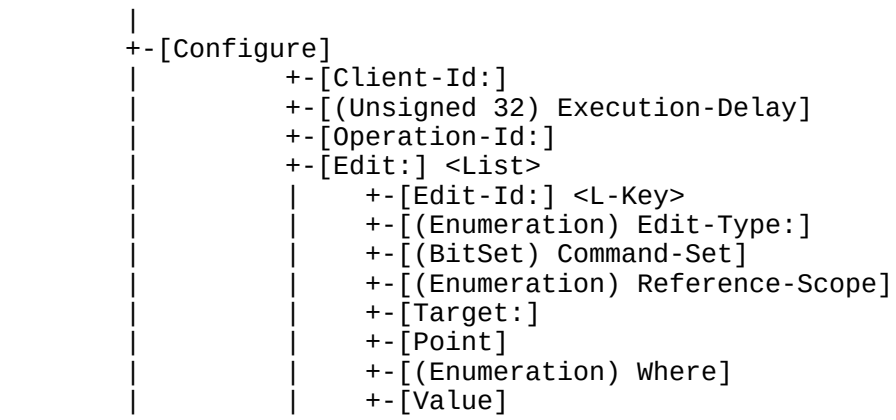


Figure 16: Configure Request

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.

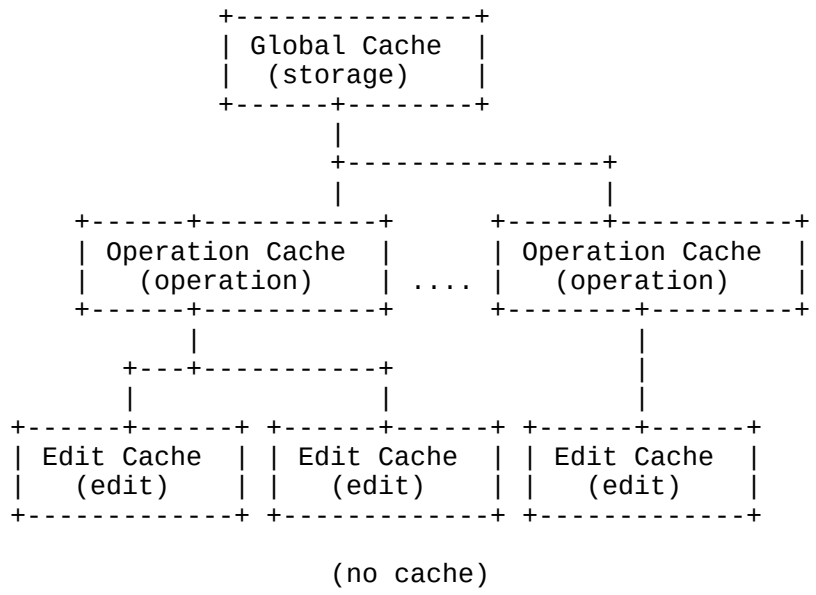


Figure 17: Example Hierarchical Cache

5.1.1.4. Operation Response

5.1.1.4.1. Immediate Response

The Response MUST include the following:

Operation Identifier of the corresponding request.

Global Status for the operation (see Table 1).

A list of Edit results (described below).

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

Edit-Id: Edit Identifier.

Edit-Status: OK.

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

Notify-Follows - A boolean indicator that the edit has been accepted by the Agent but further processing is required. A Configure-Result-Notification will be sent once the processing has succeeded or failed.

Subsequent-Edits-List: This is a list of Edits that were required to fulfill the request. It follows the edit request semantics (see Section 5.1.1.2).

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

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

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

Figure 18: Configure Operation Response

5.1.1.4.2. Asynchronous Notification

A Configure-Result-Notification occurs after the Agent has completed processing related to a Configure request. It is an asynchronous communication from the Agent to the Client.

It is identical to the immediate response with the exception that the Notify-Follows, if present, MUST be false. As this value is unnecessary it SHOULD be omitted.

5.1.1.5. Reserved Identities

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

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

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

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

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

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

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

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

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

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

"Requested-<Role>" represents requested policies that have not been assigned to a DPN and can only be assigned to DPNs that fulfill the specified Role.

It is possible to have policies in the "Requested" DPN that do not appear in other entries which reflects the inability to successfully assign the policy.

5.1.2. Monitor Messages

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

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

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

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

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

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

The Monitor messages use the following data:

Monitor-Key: Monitor Key.

Monitor: A Monitor configuration (see Section 4.9.7).

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

```
|
+-[Register-Monitor]
|         +-[Client-Id:]
|         +-[(Unsigned 32) Execution-Delay]
|         +-[Operation-Id:]
|         +-[Monitor:] <List>
|         |         +-[Extensible:]
|         |         +-[Monitor-Key:] <U-Key>
|         |         +-[Target:]
|         |         +-[Deferrable]
|         |         +-[Configuration:]

|
+-[Deregister-Monitor]
|         +-[Client-Id:]
|         +-[(Unsigned 32) Execution-Delay]
|         +-[Operation-Id:]
|         +-[Monitor:] <List>
|         |         +-[Monitor-Key:] <U-Key>
|         |         +-[(Boolean) Send-Data ~ False]

|
+-[Probe]
|         +-[Client-Id:]
|         +-[(Unsigned 32) Execution-Delay]
|         +-[Operation-Id:]
|         +-[Monitor-Key:] <List>
```

Figure 19: Monitor Messages

5.1.2.1. Asynchronous Notification

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

Triggers include but are not limited to

- 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>
        |         |    +-[Trigger:]
        |         |    +-[Monitor-Key:]
        |         |    +-[Report-Value]
```

Figure 20: Monitor Messages

5.2. Protocol Operation

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

5.2.1. DPN Selection

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

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

```
 "topology-information-model" : {
```

```
     "dpn" : [ {
      "dpn-key" : "dpn1",
      "interface" : [ {
        "interface-key" : "ifc1",
        "role" : "lma",
        "protocol" : [ "pmip" ],
        "interface-configuration" : [ {
            "index" : 0,
            "setting" : [ "optionA" : "OFF" ]
        } ]
      },{
        "interface-key" : "ifc2",
        "role" : "lma",
        "protocol" : [ "pmip" ],
        "interface-configuration" : [ {
            "index" : 0,
            "setting" : [ "optionC" : "OFF"  ]
         } ]
       },{
        "interface-key" : "ifc2-b",
        "role" : "mag",
        "protocol" : [ "pmip" ]
        } ] },
      {
      "dpn-key" : "dpn2",
      "interface" : [ {
        "interface-key" : "ifc1",
        "role" : "mag",
        "protocol" : [ "pmip" ],
        "interface-configuration" : [ {
          "index" : 0,
          "settings" : [ "optionA" : "OFF", "optionB" : "ON" ]
        } ]
      } ] }
     ],
     ...
     },
     "service-group" : [
     { "service-group-key" : "group1",
       "service-group-name" : "Anchors-OptionA-OFF",
       "role-key" : "lma",
       "protocol" : [ "pmip" ],
       "service-group-configuration" : [ {
            "index" : 0,
            "setting" : [ "optionA" : "OFF" ]
       } ],
       "dpn" : [
       { "dpn-key" : "dpn1",
```

```
        "referenced-interface" : [ { "interface-key" : "ifc1" } ] }
      ]
    },{ "service-group-key" : "group2",
      "service-group-name" : "Anchors",
      "role-role" : "lma",
      "protocol" : [ "pmip" ],
      "dpn" : [
      { "dpn-key" : "dpn1",
        "referenced-interface" : [ { "interface-key" : "ifc2" } ] }
      ]
     },{ "service-group-key" : "group3",
      "service-group-name" : "MAGs",
      "role-role" : "mag",
      "protocol" : [ "pmip" ],
      "dpn" : [
      { "dpn-key" : "dpn2",
        "referenced-interface" : [ { "interface-key" : "ifc1" } ] },
      { "dpn-key" : "dpn1",
        "referenced-interface" : [ { "interface-key" : "ifc2-b" } ] }
      ]
     }
  ]
```

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

Figure 21: Monitor Messages

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

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

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

A third group is comprised of interfaces that support the MAG function of the LMA protocol. It contains the MAG interfaces form both dpn1 and dpn2.

Given the task to find a LMA that supports the PMIP protocol the Client can determine that dpn1 is its only option and, depending on its requirement of OptionA, can appropriately determine which interface to select.

5.2.2. Policy Creation and Installation

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

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

Figure 22: Example Policy Installation (focus on FPC reference point)

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

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

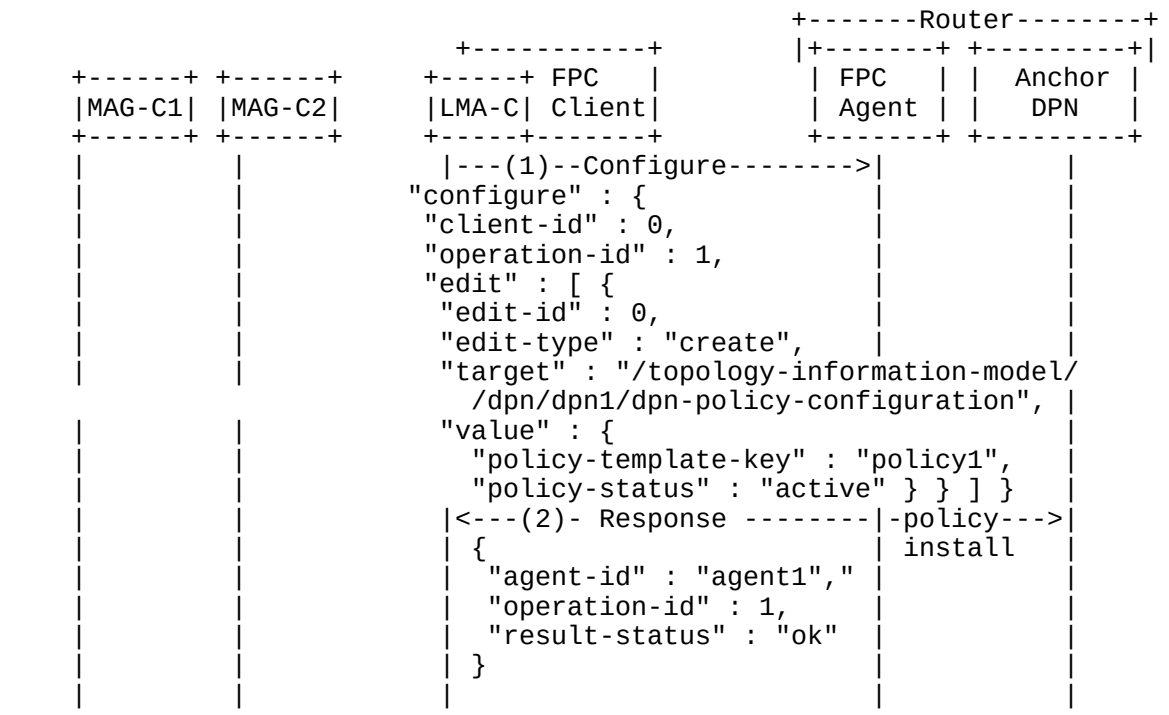


Figure 23: Example Policy Installation (focus on FPC reference point)

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

5.2.3. Simple RPC Operation

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

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

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

status, Error-Type and Error-Tag accordingly and MAY clear the entity, e.g. Mobility-Context, which caused the failure, in the response.

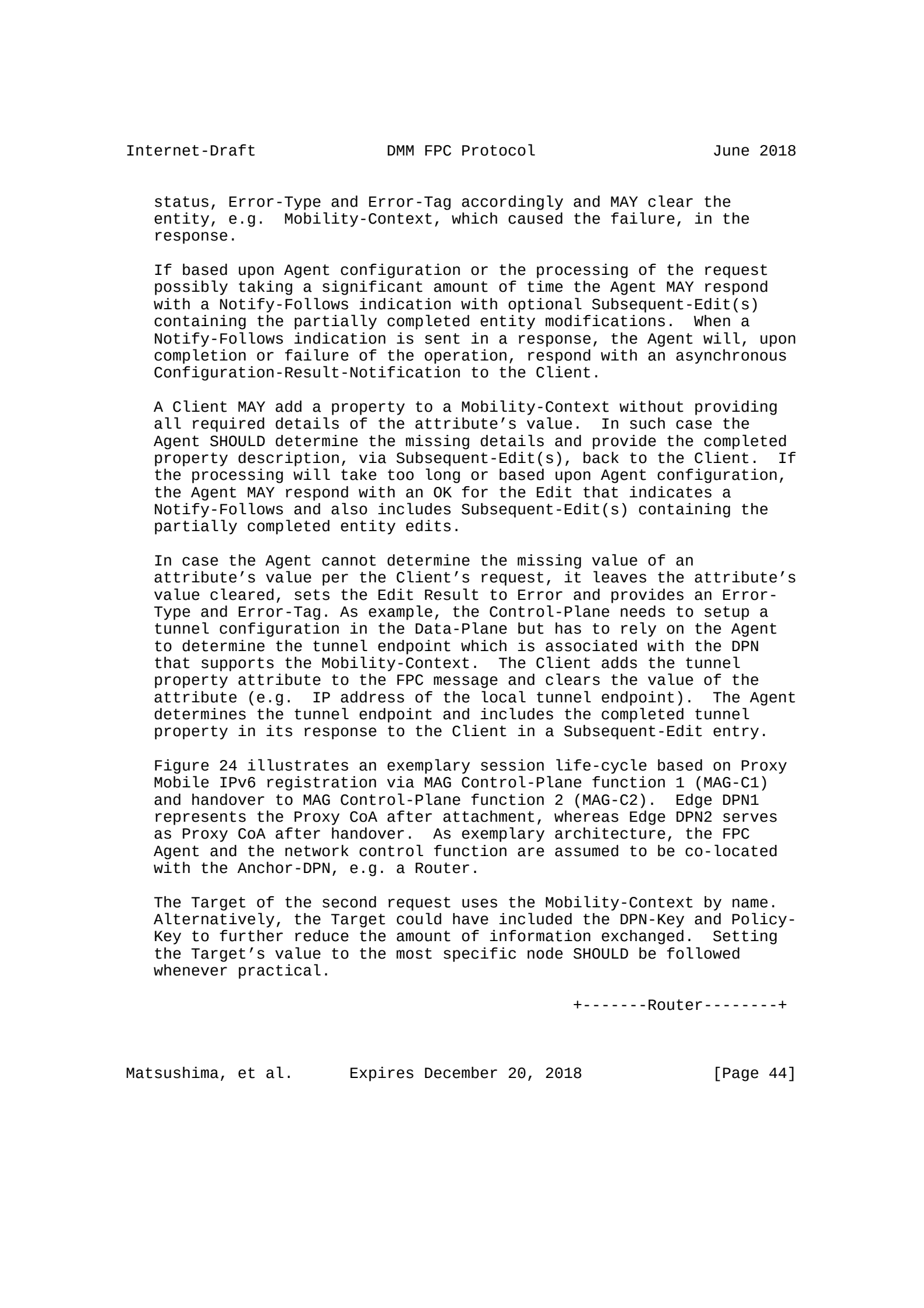
If based upon Agent configuration or the processing of the request possibly taking a significant amount of time the Agent MAY respond with a Notify-Follows indication with optional Subsequent-Edit(s) containing the partially completed entity modifications. When a Notify-Follows indication is sent in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

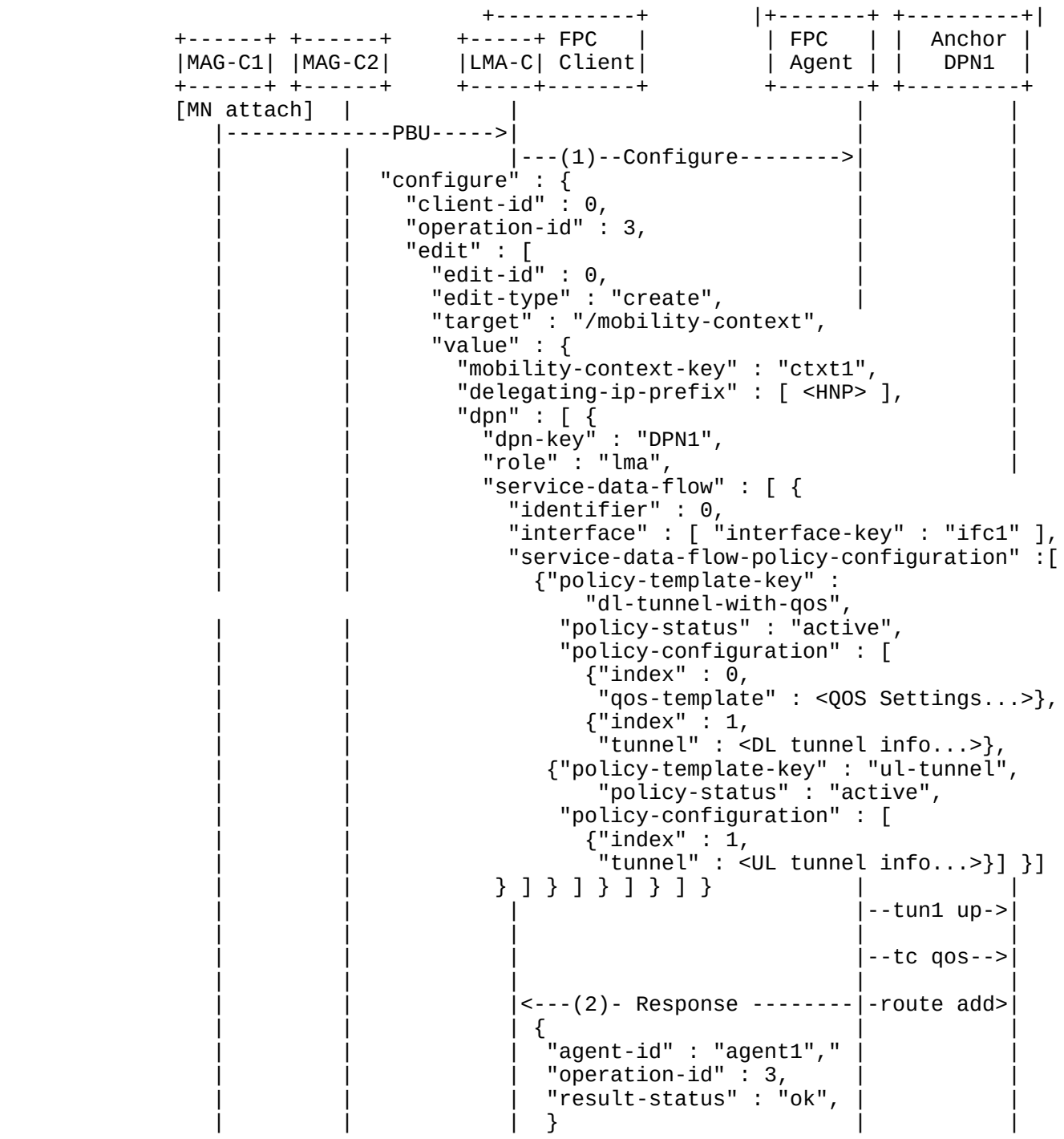
A Client MAY add a property to a Mobility-Context without providing all required details of the attribute's value. In such case the Agent SHOULD determine the missing details and provide the completed property description, via Subsequent-Edit(s), back to the Client. If the processing will take too long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

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

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

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




+-----------+ |+-------+ +---------+|
+------+ +------+ +-----+ FPC | | FPC | | Anchor |
|MAG-C1| |MAG-C2| |LMA-C| Client| | Agent | | DPN1 |
+------+ +------+ +-----+-------+ +-------+ +---------+
[MN attach] | | | | |
-------------PBU----->			
		---(1)--Configure-------->	
	"configure" : {	
	"client-id" : 0,	
	"operation-id" : 3,	
	"edit" : [	
	"edit-id" : 0,	
	"edit-type" : "create",	
	"target" : "/mobility-context",
	"value" : {
	"mobility-context-key" : "ctxt1",
	"delegating-ip-prefix" : [<HNP>],
	"dpn" : [{
	"dpn-key" : "DPN1",
	"role" : "lma",
	"service-data-flow" : [{
	"identifier" : 0,
	"interface" : ["interface-key" : "ifc1"],
	"service-data-flow-policy-configuration" :[
	{"policy-template-key" :
"dl-tunnel-with-qos",
	"policy-status" : "active",
	"policy-configuration" : [
	{"index" : 0,
"qos-template" : <QOS Settings...>},
	{"index" : 1,
"tunnel" : <DL tunnel info...>},
	{"policy-template-key" : "ul-tunnel",
"policy-status" : "active",
	"policy-configuration" : [
	{"index" : 1,
"tunnel" : <UL tunnel info...>}] }]
	}] }] }] }] }	
			--tun1 up->
			--tc qos-->
		<---(2)- Response --------	-route add>
		{	
		"agent-id" : "agent1","	
		"operation-id" : 3,	
		"result-status" : "ok",	
		}	

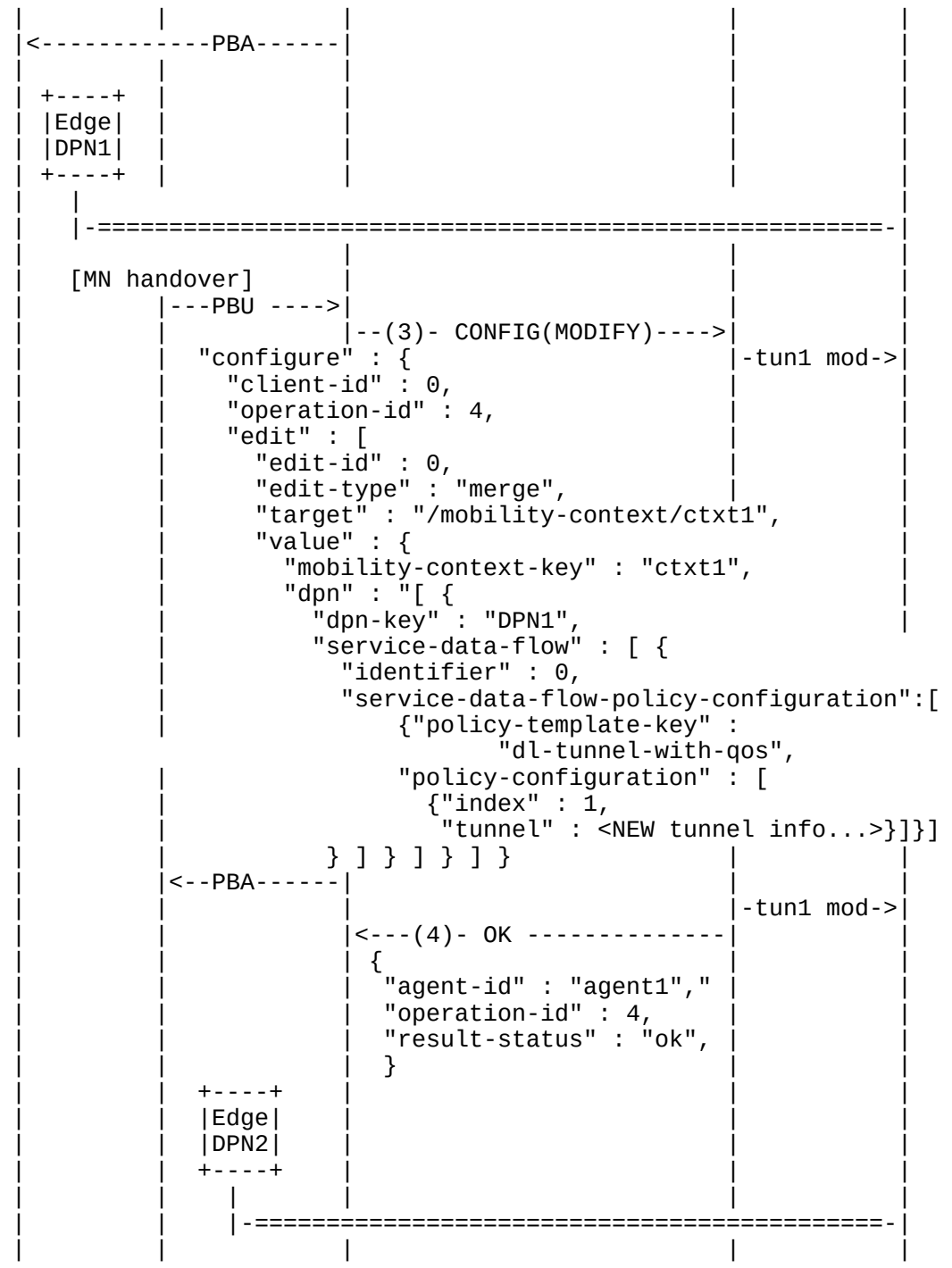

|<------------PBA------|
| +----+ |
| |Edge| |
| |DPN1| |
| +----+ |
[MN handover]
|---PBU ---->|
|--(3)- CONFIG(MODIFY)---->|
"configure" : {
"client-id" : 0,
"operation-id" : 4,
"edit" : [
"edit-id" : 0,
"edit-type" : "merge",
"target" : "/mobility-context/ctxt1",
"value" : {
"mobility-context-key" : "ctxt1",
"dpn" : "[{
"dpn-key" : "DPN1",
"service-data-flow" : [{
"identifier" : 0,
"service-data-flow-policy-configuration":[
{"policy-template-key" :
"dl-tunnel-with-qos",
"policy-configuration" : [
{"index" : 1,
"tunnel" : <NEW tunnel info...>}]}]
}] }] }] }
-tun1 mod->
|<--PBA------|
|<---(4)- OK --------------|
{
"agent-id" : "agent1","
"operation-id" : 4,
"result-status" : "ok",
}
-tun1 mod->
+----+
|Edge|
|DPN2|
+----+

Figure 24: Single Agent with Handover (focus on FPC reference point)

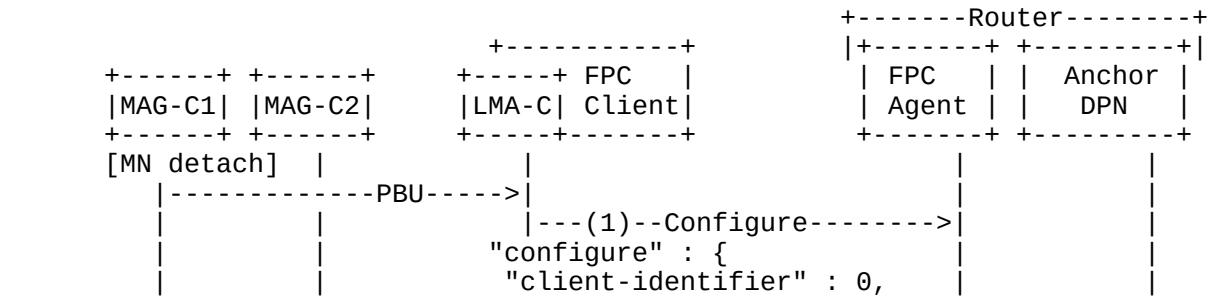
After reception of the Proxy Binding Update (PBU) at the LMA Control-Plane function (LMA-C), the LMA-C selects a suitable DPN, which serves as Data-Plane anchor to the mobile node's (MN) traffic. The LMA-C adds a new logical Mobility-Context to the DPN to treat the MN's traffic (1) and includes a Mobility-Context-Key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

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

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

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

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



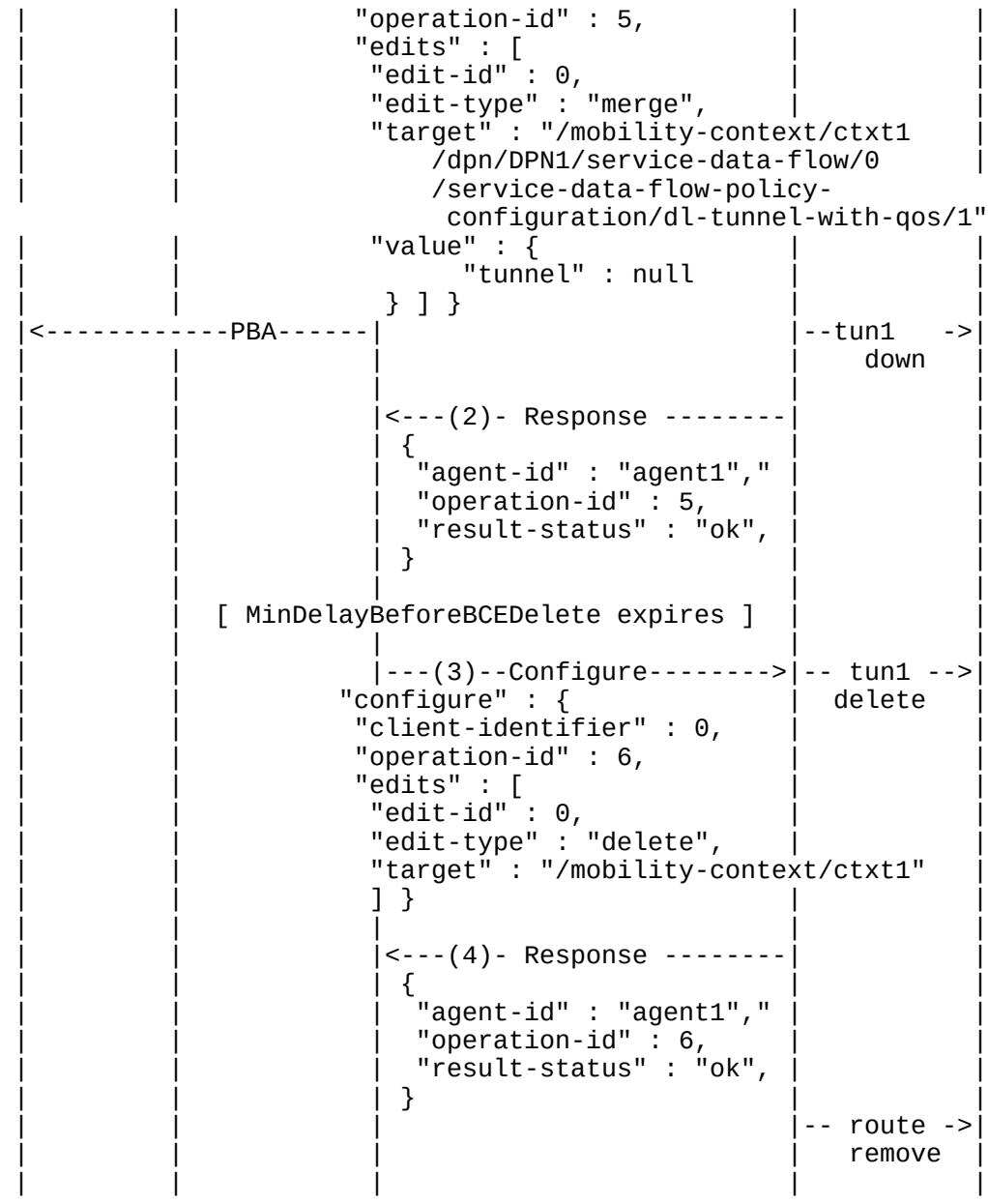


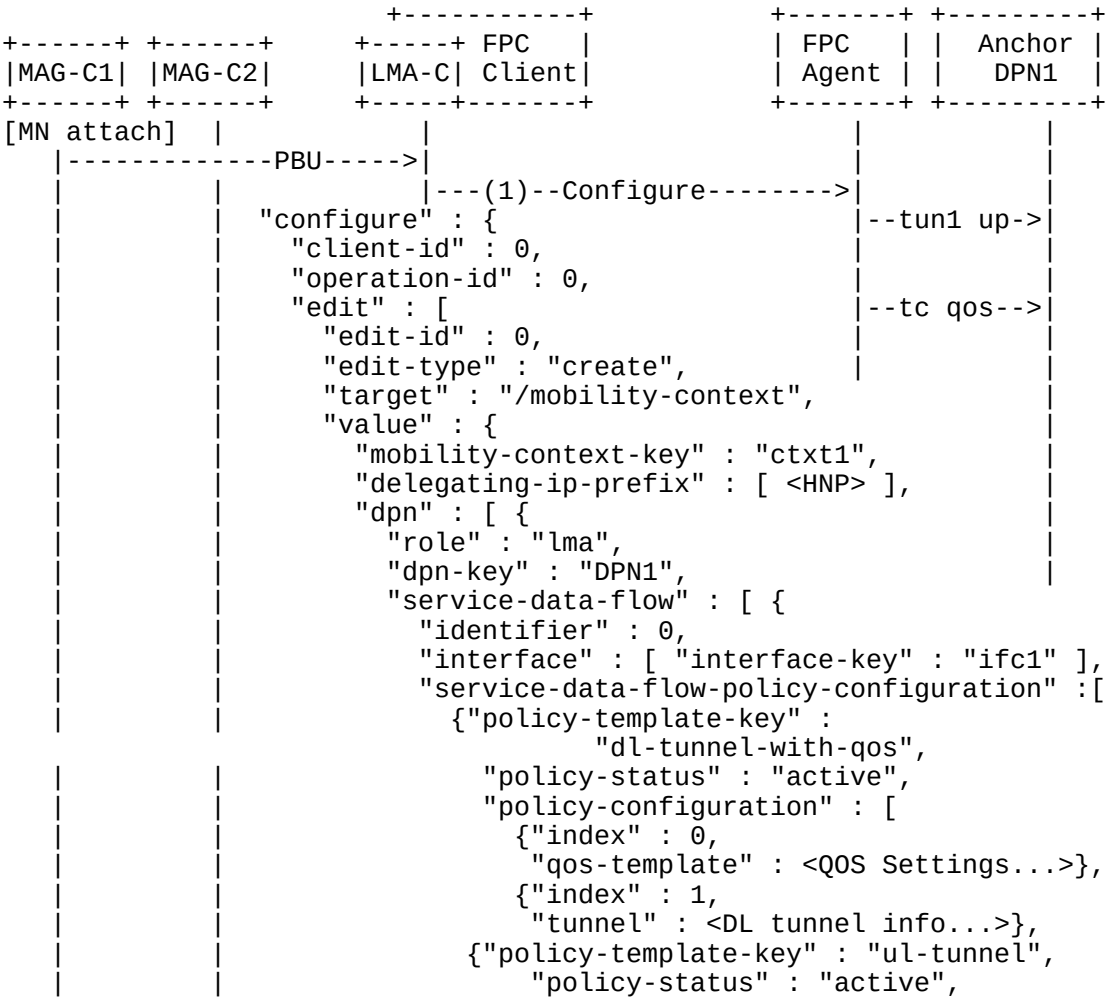
Figure 25: Single Agent with Deletion (focus on FPC reference point)

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

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

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

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




"policy-configuration" : [
{"index" : 1,
"tunnel" : <UL tunnel info...>}] }]
}] }] }, {
"dpn-key" : "DPN2",
"role" : "mag",
"service-data-flow" : [{
"identifier" : 0,
"interface" : ["interface-key" : "ifc2"],
"service-data-flow-policy-configuration" :[
{"policy-template-key" :
"dl-tunnel-with-qos",
"policy-status" : "active",
"policy-configuration" : [
{"index" : 0,
"qos-template" : <QOS Settings...>},
{"index" : 1,
"tunnel" : <DL tunnel info...>},
{"policy-template-key" : "ul-tunnel",
"policy-status" : "active",
"policy-configuration" : [
{"index" : 1,
"tunnel" : <UL tunnel info...>}] }]
}] }] }
] }] }
<---(2)- Response --------
{
"agent-id" : "agent1","
"operation-id" : 0,
"result-status" : "ok",
"notify-follows" : "true",
}
-route add>
PBA------
Edge
DPN2
tun1 up -------------
tc qos --------------
route add -----------
(3) Configure-Result-
Notification
{
"agent-id" : "agent1","
-route add>

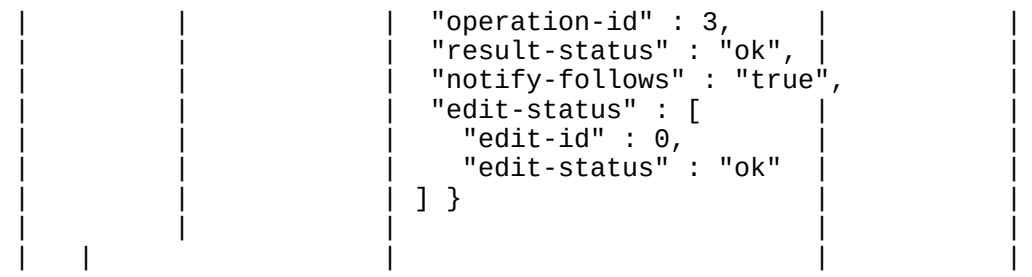


Figure 26: Exemplary Message Sequence for Multi-DPN Agent

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.

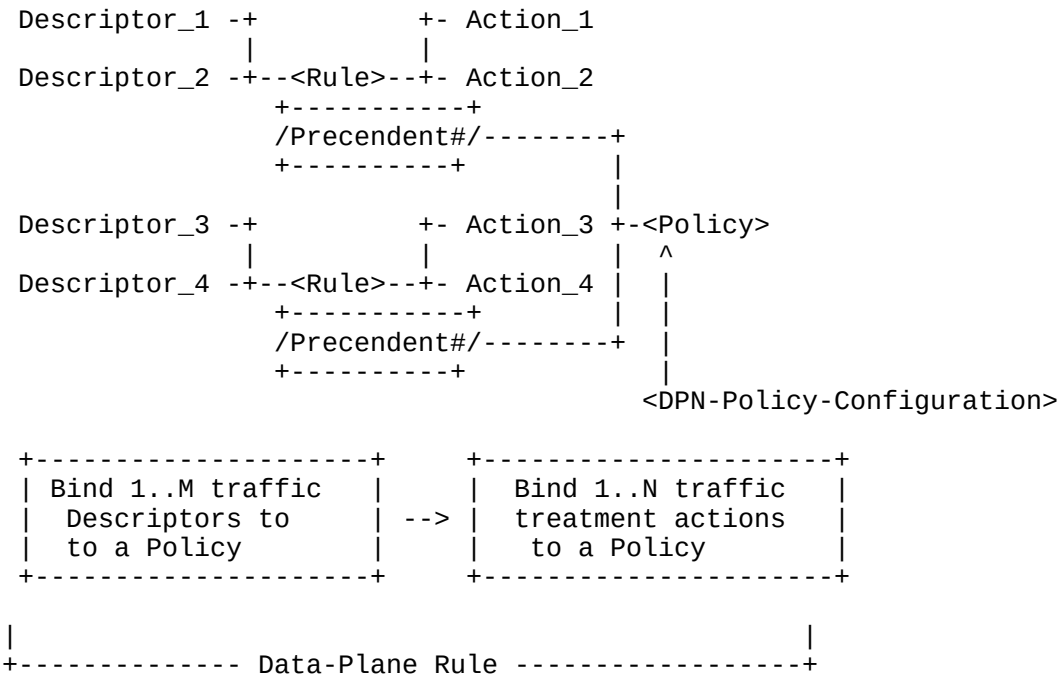


Figure 27: Structure of Configurable Policies

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

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

Figure 28 illustrates the generic context configuration model as used between a Client and an Agent.

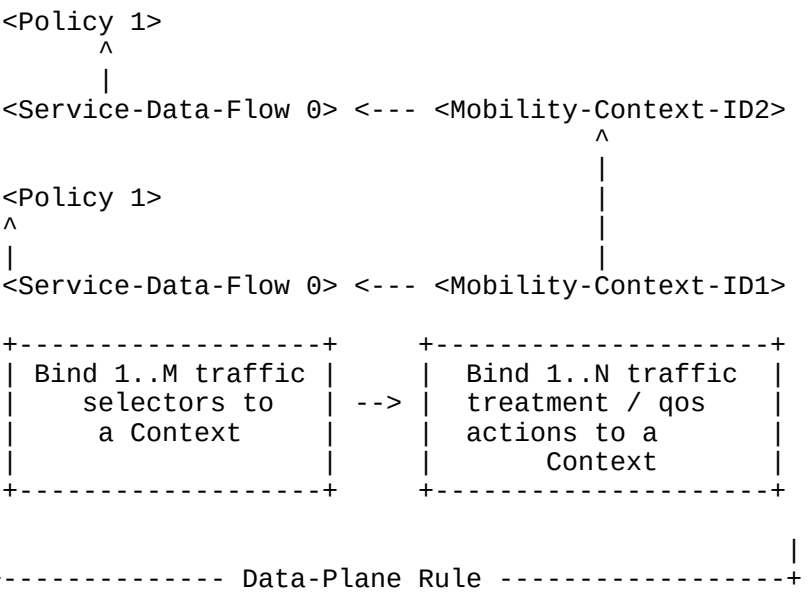


Figure 28: Mobility Context Hierarchy

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

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

5.2.5. Monitor Example

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

Note, specific event identifiers and types are out of scope.

Figure 29: Monitor Example (focus on FPC reference point)

6. Templates and Command Sets

Configuration templates are shown below.

6.1. Monitor Configuration Templates

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

A scheduled configuration specifies a time for reporting.

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

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

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

Figure 30: Monitor Configuration Templates

6.2. Descriptor Templates

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

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

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

```
  |
  +-[ip-prefix-template]
  |       +-[(IP Prefix / Length) To-IP-Prefix]
  |       +-[(IP Prefix / Length) From-IP-Prefix]
  ...
  +-[pmip-traffic-selector]
```

```
|       +-[(Enumerated - IPv4 or IPv6) ts-format]
|       +-[ipsec-spi-range]
|       |   +-[ (ipsec-spi) start-spi: ]
|       |   +-[ (ipsec-spi) end-spi ]
|       +-[source-port-range]
|       |   +-[ (port-number) start-port: ]
|       |   +-[ (port-number) end-port ]
|       +-[destination-port-range]
|       |   +-[ (port-number) start-port: ]
|       |   +-[ (port-number) end-port ]
|       +-[source-address-range-v4]
|       |   +-[ (ipv4-address) start-address: ]
|       |   +-[ (ipv4-address) end-address ]
|       +-[destination-address-range-v4]
|       |   +-[ (ipv4-address) start-address: ]
|       |   +-[ (ipv4-address) end-address ]
|       +-[ds-range]
|       |   +-[ (dscp) start-ds: ]
|       |   +-[ (dscp) end-ds ]
|       +-[protocol-range]
|       |   +-[ (uint8) start-protocol: ]
|       |   +-[ (uint8) end-protocol ]
|       +-[source-address-range-v6]
|       |   +-[(ipv6-address) start-address: ]
|       |   +-[(ipv6-address) end-address ]
|       +-[destination-address-range-v6]
|       |   +-[(ipv6-address) start-address: ]
|       |   +-[(ipv6-address) end-address ]
|       +-[flow-label-range]
|       |   +-[(ipv6-flow-label) start-flow-label ]
|       |   +-[(ipv6-flow-label) end-flow-label ]
|       +-[traffic-class-range]
|       |   +-[ (dscp) start-traffic-class ]
|       |   +-[ (dscp) end-traffic-class ]
|       +-[next-header-range]
|       |   +-[ (uint8) start-next-header ]
|       |   +-[ (uint8) end-next-header ]
|   ...
|   +-[rfc5777-classifier]
|       +-[Extensible: True]
|       +-[(uint8) protocol]
|       +-[(Enumerated - In/Out/Both) Direction]
|       +-[From-Spec] <List>
|       |    +-[(ip-address) IP-Address] <List>
|       |    +-[IP-Address-Range] <List>
|       |    |    +-[(ip-address) IP-Address-Start]
|       |    |    +-[(ip-address) IP-Address-End]
|       |    +-[IP-Address-Mask] <List>
```

```
|       |    |    +-[(ip-address) IP-Address:]
|       |    |    +-[(Unsigned 32) IP-Bit-Mask-Width:]
|       |    +-[(mac-address) MAC-Address] <List>
|       |    +-[MAC-Address-Mask] <List>
|       |    |    +-[(mac-address) MAC-Address:]
|       |    |    +-[(mac-address) MAC-Address-Mask-Pattern:]
|       |    +-[(eui64-address) EUI64-Address] <List>
|       |    +-[EUI64-Address-Mask] <List>
|       |    |    +-[(eui64-address) EUI64-Address:]
|       |    |    +-[(eui64-address) EUI64-Address-Mask-Pattern:]
|       |    +-[(Integer 32) Port] <List>
|       |    +-[Port-Range] <List>
|       |    |    +-[(Integer 32) Port-Start]
|       |    |    +-[(Integer 32) Port-End]
|       |    +-[(Boolean) Negated]
|       |    +-[(Boolean) Use-Assigned-Address]
|       +-[To-Spec] <List> (O)
|       |    +-[(ip-address) IP-Address] <List>
|       |    +-[IP-Address-Range] <List>
|       |    |    +-[(ip-address) IP-Address-Start]
|       |    |    +-[(ip-address) IP-Address-End]
|       |    +-[IP-Address-Mask] <List>
|       |    |    +-[(ip-address) IP-Address:]
|       |    |    +-[(Unsigned 32) IP-Bit-Mask-Width:]
|       |    +-[(mac-address) MAC-Address] <List>
|       |    +-[MAC-Address-Mask] <List>
|       |    |    +-[(mac-address) MAC-Address:]
|       |    |    +-[(mac-address) MAC-Address-Mask-Pattern:]
|       |    +-[(eui64-address) EUI64-Address] <List>
|       |    +-[EUI64-Address-Mask] <List>
|       |    |    +-[(eui64-address) EUI64-Address:]
|       |    |    +-[(eui64-address) EUI64-Address-Mask-Pattern:]
|       |    +-[(Integer 32) Port] <List>
|       |    +-[Port-Range] <List>
|       |    |    +-[(Integer 32) Port-Start]
|       |    |    +-[(Integer 32) Port-End]
|       |    +-[(Boolean) Negated]
|       |    +-[(Boolean) Use-Assigned-Address]
|       +-[(dscp) Diffserv-Code-Point] <List>
|       +-[(Boolean) Fragmentation-Flag ~ False]
|       +-[IP-Option] <List>
|       +-[TCP-Option] <List>
|       +-[TCP-Flags]
|       +-[ICMP-Type] <List>
|       +-[ETH-Option] <List>
|       +-[ecn-ip-codepoint] <List>
|       +-[(flowlabel) flow-label] <List>
|       +-[flow-label-range] <List>
```

```
     |       |    +-[(flowlabel) flow-label-start]
     |       |    +-[(flowlabel) flow-label-end]
```

Figure 31: Descriptor Templates

6.3. Tunnel Templates

The Network Service Header is specified in [RFC8300].

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

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

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

For GRE, the gre-key MUST be present.

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

local tunnel endpoint identifier (teid) - MUST be present if local-address is nonempty

remote tunnel endpoint identifier (teid) - MUST be present if remote-address is nonempty

sequence-numbers-on - Indicates that sequence numbers will be used

Tunnels can be used as Next Hop and Descriptor values.

```
     |
     +-[next-hop-template]
     |     +-[Extensible: True]
     |     +-[(ip-address) address]
     |     +-[(mac-address) mac-address]
     |     +-[(service-path-id) service-path]
     |     +-[(mpls-label) mpls-path]
     |     +-[(network service header) nsh]
     |     +-[(Unsigned Integer) interface]
     |     +-[(Unsigned 128) segment-identifier]
     |     +-[(MPLS Stack) mpls-label-stack]
     |     +-[(MPLS SR Stack) mpls-sr-stack]
     |     +-[(IPv6 SR Stack) srv6-stack]
     |     +-[tunnel-template]
     ...
     |
     +-[tunnel-template]
     |     +-[Extensible: True]
     |     +-[(address) local-address]
     |     +-[(address) remote-address]
     |     +-[mtu]
     |     +-[(Enumeration - ipv4(0), ipv6(1), dual(2) payload_type:]
     |     +-[(Enumeration - ip-in-ip(0),
                    udp(1), gre(2), gtpv1(3), gtpv2(4)) type:]
     |     +-[interface]
     |     +-[next-hop]
     |     +-[gre-key:] (type == gre)
     |     +-[gtp-info] (type == gtpv1 or type == gtpv2 )
     |     |    +-[(Unsigned 32) local-teid]
     |     |    +-[(Unsigned 32) remote-teid]
     |     |    +-[(Boolean) sequence-numbers-on] (type == gtpv1)
```

Figure 32: Tunnel Templates

6.4. Action Templates

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

Drop action has no values.

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

Copy-Forward creates a copy of the packet and then forwards it in accordance to the nexthop value.

```
                    |
                    +-[drop-template]
                    ...
                    |
                    +-[rewrite-template]
                    |     +-[Extensible: True]
                    |     +-[ip-prefix-template]
                    |     +-[pmip-traffic-selector]
                    |     +-[rfc5777-classifier]
                    ...
                    |
                    +-[copy-forward-template]
                    |     +-[Extensible: True]
                    |     +-[next-hop:]
```

Figure 33: Action Templates

6.5. Quality of Service Action Templates

PMIP QoS is specified in [RFC7222].

```
    |
    +-[qos-template]
    |     +-[Extensible: True]
    |     +-[(dscp) trafficclass]
    |     +-[pmip-qos]
    |     |     +-[(Unsigned 32) per-mn-agg-max-dl]
    |     |     +-[(Unsigned 32) per-mn-agg-max-ul]
    |     |     +-[per-session-agg-max-dl]
    |     |     |    +-[(Unsigned 32) max-rate:]
    |     |     |    +-[(Boolean) service-flag:]
    |     |     |    +-[(Boolean) exclude-flag:]
    |     |     +-[per-session-agg-max-ul]
    |     |     |    +-[(Unsigned 32) max-rate:]
    |     |     |    +-[(Boolean) service-flag:]
    |     |     |    +-[(Boolean) exclude-flag:]
    |     |     +-[allocation-retention-priority]
    |     |     |    +-[(Unsigned 8) priority-level:]
    |     |     |    +-[(Enumeration) preemption-capability:]
    |     |     |    +-[(Enumeration) preemption-vulnerability:]
    |     |     +-[(Unsigned 32) agg-max-dl]
    |     |     +-[(Unsigned 32) agg-max-ul]
    |     |     +-[(Unsigned 32) gbr-dl]
    |     |     +-[(Unsigned 32) gbr-ul]
```

Figure 34: QoS Templates

6.6. PMIP Command-Set

The following Command Set values are supported for IETF PMIP.

- 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.

```
                    |
                    +-[ MN-Policy-Template ]
                    |     +-[(Unsigned 64) imsi:]
                    ...
                    +-[tunnel-template]
                    |     +-[Extensible: True]
                    |     +-[(unsigned 4) ebi:]
                    |     +-[(unsigned 4) lbi]
                    ...
                    +-[qos-template]
                    |     +-[Extensible: True]
                    |     +-[(unsigned 4) qos-class-identifier]
                    |     +-[(Unsigned 32) ue-agg-max-bitrate]
                    |     +-[(Unsigned 32) apn-agg-max-bitrate]
                    ...
```

Figure 35: 3GPP Mobility Templates

```
|
+-[ packet-filter ]
|     +-[Extensible: True]
|     +-[(Unsigned 8) identifier:]
|     +-[Contents:] <List>
|     |     +-[(ip-address) ipv4-ipv6-local]
|     |     +-[(ipv6-prefix) ipv6-prefix-local]
|     |     +-[(ip-address) ipv4-ipv6-remote]
|     |     +-[(ipv6-prefix) ipv6-prefix-remote]
|     |     +-[(Unsigned 8) protocol-next-header]
|     |     +-[(Unsigned 16) local-port]
|     |     +-[local-port-range]
|     |     |      +-[(Unsigned 16) local-port-lo]
|     |     |      +-[(Unsigned 16) local-port-hi]
|     |     +-[(Unsigned 16) remote-port]
|     |     +-[remote-port-range]
|     |     |      +-[(Unsigned 16) remote-port-lo]
|     |     |      +-[(Unsigned 16) remote-port-hi]
|     |     +-[(Unsigned 32) sec-parameter-index]
|     |     +-[(dscp) traffic-class]
|     |     +-[traffic-class-range]
|     |     |      +-[(dscp) traffic-class-lo]
|     |     |      +-[(dscp) traffic-class-hi]
|     |     +-[(dscp) flow-label]
...
```

Figure 36: 3GPP Packet Filter Template (Descriptor)

The following Command Set values are supported for 3GPP.

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

7. Implementation Status

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

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

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

A throughput performance of tens per second using various NetConf based solutions in OpenDaylight made fpcagent, based on version 03, undesirable for call processing. The RPC implementation improved throughput by an order of magnitude but was not useful based upon FPC's Version 03 design using two information models. During this time the features of version 04 and its converged model became attractive and the fpcagent project was closed in August 2016.

fpcagent will no longer be developed and will remain a proprietary implementation.

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

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

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

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

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

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

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

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

fpc currently supports the following:

1 proprietary DPN API

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

CONFIG and CONF_BUNDLE (all operations)

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

Immediate Response is always an OK_NOTIFY_FOLLOWS.

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

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

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

Figure 37: fpc pseudo code

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

NOTE: Tenant support requires binding a Client ID to a Tenant ID (it is a one to many relation) but that is outside of the scope of this

specification. Otherwise, the specification is complete in terms of providing sufficient information to implement an Agent.

8. Security Considerations

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

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

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

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

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

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

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

Some of the readable data nodes defined may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to

these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

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

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

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

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

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

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

General usage of FPC MUST consider the following:

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

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

9. IANA Considerations

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

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

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

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

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

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

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

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

      name:         ietf-dmm-pmip-qos
      namespace:    urn:ietf:params:xml:ns:yang:ietf-dmm-pmip-qos
      prefix:       qos-pmip
      reference:    TBD2

      name:         ietf-dmm-traffic-selector-types
      namespace:    urn:ietf:params:xml:ns:yang:
        ietf-dmm-traffic-selector-types
      prefix:       traffic-selectors
      reference:    TBD3

      name:         ietf-dmm-fpc-settingsext
      namespace:    urn:ietf:params:xml:ns:yang:
        ietf-dmm-fpc-settingsext
      prefix:       fpcbase
      reference:    TBD4

        name:         ietf-diam-trafficclassifier
        namespace:    urn:ietf:params:xml:ns:yang:
        ietf-diam-trafficclassifier
        prefix:       diamclassifier
        reference:    TBD5
```

10. Work Team Participants

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

11. References

11.1. Normative References

[I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-13 (work in progress), May 2018.

[I-D.ietf-spring-segment-routing-mpls]
Bashandy, A., Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-14 (work in progress), June 2018.

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

[RFC5777] Korhonen, J., Tschofenig, H., Arumaithurai, M., Jones, M., Ed., and A. Lior, "Traffic Classification and Quality of Service (QoS) Attributes for Diameter", RFC 5777, DOI 10.17487/RFC5777, February 2010, <https://www.rfc-editor.org/info/rfc5777>.

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

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

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

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8072] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Patch Media Type", RFC 8072, DOI 10.17487/RFC8072, February 2017, <https://www.rfc-editor.org/info/rfc8072>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

11.2. Informative References

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

[I-D.ietf-dmm-deployment-models] Gundavelli, S. and S. Jeon, "DMM Deployment Models and Architectural Considerations", draft-ietf-dmm-deployment-models-04 (work in progress), May 2018.

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

[RFC3958] Daigle, L. and A. Newton, "Domain-Based Application Service Location Using SRV RRs and the Dynamic Delegation Discovery Service (DDDS)", RFC 3958, DOI 10.17487/RFC3958, January 2005, <https://www.rfc-editor.org/info/rfc3958>.

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

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

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

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

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

[RFC7660] Bertz, L., Manning, S., and B. Hirschman, "Diameter Congestion and Filter Attributes", RFC 7660, DOI 10.17487/RFC7660, October 2015, <https://www.rfc-editor.org/info/rfc7660>.

Appendix A. YANG Data Model for the FPC protocol

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

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

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

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

Mapping of templates to YANG are performed as follows:

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

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

The CONFIGURE and CONFIGURE-RESULT-NOTIFICATION use the yang-patch-status which is a container for edits. This was done to maximize YANG reuse.

In the configure rpc, operation-id is mapped to patch-id and in an edit the edit-type is mapped to operation.

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

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

Five modules are defined:

- 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)

- qosattribute (ietf-qos-pmip module)
- qosoption (ietf-qos-pmip module)
- Allocation-Retention-Priority-Value (ietf-qos-pmip module)
- trafficselector (ietf-traffic-selector-types)
- classifier (ietf-diam-trafficclassifier)
- packet-filter (ietf-dmm-fpc-settingsext)
- instructions (ietf-dmm-fpc-settingsext)
- fpc-descriptor-value (ietf-dmm-fpc-settingsext)
- fpc-action-value (ietf-dmm-fpc-settingsext)

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

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

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

A.1. FPC YANG Model

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

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

```
<CODE BEGINS> file "ietf-dmm-fpc@2018-05-17.yang"
module ietf-dmm-fpc {
  yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-fpc";
    prefix fpc;
```

```
       import ietf-inet-types { prefix inet;
           revision-date 2013-07-15; }
       import ietf-dmm-fpc-settingsext { prefix fpcbase;
           revision-date 2018-05-17; }
       import ietf-diam-trafficclassifier { prefix rfc5777;
           revision-date 2018-05-17; }
       import ietf-restconf { prefix rc;
           revision-date 2017-01-26; }
       import ietf-yang-patch { prefix ypatch;
           revision-date 2017-02-22; }

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

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

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

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

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

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

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

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

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

       revision 2018-05-17 {
```

```
      description "Initial Revision.";
      reference "draft-ietf-dmm-fpc-cpdp-10";
    }

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

```
            type string;
        }
        description "FPC Identity";
    }
    grouping index {
        leaf index {
            type uint16;
            description "Index";
        }
        description "Index Value";
    }

    // Policy Structures
    grouping descriptor-template-key {
        leaf descriptor-template-key {
            type fpc:fpc-identity;
            mandatory true;
            description "Descriptor Key";
        }
        description "Descriptor-Template Key";
    }
    grouping action-template-key {
        leaf action-template-key {
            type fpc:fpc-identity;
            mandatory true;
            description "Action Key";
        }
        description "Action-Template Key";
    }
    grouping rule-template-key {
        leaf rule-template-key {
            type fpc:fpc-identity;
            mandatory true;
            description "Rule Identifier";
        }
        description "Rule Key";
    }
    grouping policy-template-key {
        leaf policy-template-key {
            type fpc:fpc-identity;
            mandatory true;
            description "Rule Identifier";
        }
        description "Rule Key";
    }

    grouping fpc-setting-value {
        anydata setting;
```

```
            description "FPC Setting Value";
        }
        // Configuration / Settings
        grouping policy-configuration-choice {
               choice policy-configuration-value {
                    case descriptor-value {
                        uses fpcbase:fpc-descriptor-value;
                        description "Descriptor Value";
                    }
                    case action-value {
                        uses fpcbase:fpc-action-value;
                        description "Action Value";
                    }
                    case setting-value {
                        uses fpc:fpc-setting-value;
                        description "Setting";
                    }
                    description "Policy Attributes";
                }
                description "Policy Configuration Value Choice";
        }
        grouping policy-configuration {
            list policy-configuration {
                key index;
                uses fpc:index;
                uses fpc:policy-configuration-choice;
                description "Policy Configuration";
            }
            description "Policy Configuration Value";
        }
        grouping ref-configuration {
            uses fpc:policy-template-key;
            uses fpc:policy-configuration;
            uses fpc:templatedef;
            description "Policy-Configuration Entry";
        }

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

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

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

        // Topology Information Model
        identity role {
            description "Role";
        }
        grouping dpn-key {
            leaf dpn-key {
                type fpc:fpc-identity;
                description "DPN Key";
            }
            description "DPN Key";
```

```
        }
        grouping role-key {
            leaf role-key {
                type identityref {
                    base "fpc:role";
                }
                mandatory true;
                description "Access Technology Role";
            }
            description "Access Technology Role key";
        }
        grouping interface-key {
            leaf interface-key{
                type fpc:fpc-identity;
                mandatory true;
                description "interface identifier";
            }
            description "Interface Identifier key";
        }
        identity interface-protocols {
            description "Protocol supported by the interface";
        }
        identity features {
            description "Protocol features";
        }

      // Mobility Context
      grouping mobility-context {
        leaf mobility-context-key {
          type fpc:fpc-identity;
          mandatory true;
          description "Mobility Context Key";
        }
        leaf-list delegating-ip-prefix {
            type inet:ip-prefix;
            description "IP Prefix";
        }
        leaf parent-context {
          type fpc:fpc-identity;
          description "Parent Mobility Context";
        }
        leaf-list child-context {
            type fpc:fpc-identity;
            description "Child Mobility Context";
        }
        container mobile-node {
            leaf-list ip-address {
                type inet:ip-address;
```

```
                description "IP Address";
            }
            leaf imsi {
                type fpcbase:imsi-type;
                description "IMSI";
            }
            list mn-policy-configuration {
                key policy-template-key;
                uses fpc:ref-configuration;
                description "MN Policy Configuration";
            }
            description "Mobile Node";
        }
        container domain {
            leaf domain-key {
                type fpc:fpc-identity;
                description "Domain Key";
            }
            list domain-policy-settings {
                key policy-template-key;
                uses fpc:ref-configuration;
                description "MN Policy Configuration";
            }
            description "Domain";
        }
        list dpn {
            key dpn-key;
            uses fpc:dpn-key;
            list dpn-policy-configuration {
                key policy-template-key;
                uses fpc:ref-configuration;
                description "DPN Policy Configuration";
            }
            leaf role {
                type identityref {
                    base "fpc:role";
                }
                description "Role";
            }
            list service-data-flow {
                key identifier;
                leaf identifier {
                    type uint32;
                    description "Generic Identifier";
                }
                leaf service-group-key {
                    type fpc:fpc-identity;
                    description "Service Group Key";
```

```
                }
                list interface {
                    key interface-key;
                    uses fpc:interface-key;
                    description "interface assigned";
                }
                list service-data-flow-policy-configuration {
                    key policy-template-key;
                    uses fpc:ref-configuration;
                    description "Flow Policy Configuration";
                }
                description "Service Dataflow";
            }
            description "DPN";
        }
        description "Mobility Context";
      }

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

```
          leaf period {
              type uint32;
              description "Period";
          }
          case threshold-config {
            leaf low {
              type uint32;
              description "low threshold";
            }
            leaf hi {
              type uint32;
              description "high threshold";
            }
            description "Threshold Config Case";
          }
          leaf schedule {
              type uint32;
              description "Reporting Time";
          }
          leaf-list event-identities {
              type identityref {
                base "fpc:event-type";
              }
              description "Event Identities";
          }
          leaf-list event-ids {
              type uint32;
                description "Event IDs";
          }
          description "Event Config Value";
        }
        description "Monitor Configuration";
      }

      // Top Level Structures
      list tenant {
       key "tenant-key";
       leaf tenant-key {
            type fpc:fpc-identity;
            description "Tenant Key";
       }
       container topology-information-model {
         config false;
         list service-group {
            key "service-group-key role-key";
            leaf service-group-key {
                type fpc:fpc-identity;
                mandatory true;
```

```
            description "Service Group Key";
        }
        leaf service-group-name {
            type string;
            description "Service Group Name";
        }
        uses fpc:role-key;
        leaf role-name {
            type string;
            mandatory true;
            description "Role Name";
        }
        leaf-list protocol {
          type identityref {
            base "interface-protocols";
          }
          min-elements 1;
          description "Supported protocols";
        }
        leaf-list feature {
            type identityref {
                base "interface-protocols";
            }
            description "Supported features";
        }
        list service-group-configuration {
            key index;
            uses fpc:index;
            uses fpc:policy-configuration-choice;
            description "Settings";
        }
        list dpn {
            key dpn-key;
            uses fpc:dpn-key;
            min-elements 1;
            list referenced-interface {
                key interface-key;
                uses fpc:interface-key;
                leaf-list peer-service-group-key {
                    type fpc:fpc-identity;
                    description "Peer Service Group";
                }
                description "Referenced Interface";
            }
            description "DPN";
        }
        description "Service Group";
    }
```

```
       list dpn {
         key dpn-key;
         uses fpc:dpn-key;
         leaf dpn-name {
           type string;
           description "DPN name";
         }
         leaf dpn-resource-mapping-reference {
           type string;
           description "Reference to underlying DPN resource(s)";
         }
         leaf domain-key {
           type fpc:fpc-identity;
           description "Domains";
         }
         leaf-list service-group-key {
           type fpc:fpc-identity;
           description "Service Group";
         }
         list interface {
           key "interface-key";
           uses fpc:interface-key;
           leaf interface-name {
               type string;
               description "Service Endpoint Interface Name";
           }
           leaf role {
               type identityref {
                   base "fpc:role";
               }
               description "Roles supported";
           }
           leaf-list protocol {
             type identityref {
               base "interface-protocols";
             }
             description "Supported protocols";
           }
           list interface-configuration {
            key index;
            uses fpc:index;
            uses fpc:policy-configuration-choice;
            description "Interface settings";
          }
          description "DPN interfaces";
         }
         list dpn-policy-configuration {
           key policy-template-key;
```

```
            uses fpc:ref-configuration;
            description "DPN Policy Configuration";
          }
          description "Set of DPNs";
        }
        list domain {
          key domain-key;
          leaf domain-key {
            type fpc:fpc-identity;
            mandatory true;
            description "Domain Key";
          }
          leaf domain-name {
            type string;
            description "Domain displayname";
          }
          list domain-policy-configuration {
            key policy-template-key;
            uses fpc:ref-configuration;
            description "Domain Configuration";
          }
          description "List of Domains";
        }
        container dpn-checkpoint {
            uses fpc:basename-info;
            description "DPN Checkpoint information";
        }
        container service-group-checkpoint {
            uses fpc:basename-info;
            description "Service Group Checkpoint information";
        }
        container domain-checkpoint {
            uses fpc:basename-info;
            description "Domain Checkpoint information";
        }
        description "FPC Topology grouping";
      }
      container policy-information-model {
           config false;
           uses fpc:policy-information-model;
           uses fpc:basename-info;
           description "Policy";
      }
      list mobility-context {
           key "mobility-context-key";
           config false;
           uses fpc:mobility-context;
           description "Mobility Context";
```

```
      }
      list monitor {
           key monitor-key;
           config false;
           uses fpc:monitor-config;
           description "Monitor";
      }
      description "Tenant";
     }

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

       // RPCs
       grouping client-id {
           leaf client-id {
               type fpc:client-identifier;
               mandatory true;
               description "Client Id";
           }
           description "Client Identifier";
       }
       grouping execution-delay {
           leaf execution-delay {
               type uint32;
               description "Execution Delay (ms)";
           }
           description "Execution Delay";
       }
       typedef ref-scope {
         type enumeration {
```

```
            enum none {
              value 0;
              description "no references";
            }
            enum op {
              value 1;
              description "All references are intra-operation";
            }
            enum bundle {
              value 2;
              description "All references in exist in bundle";
            }
            enum storage {
              value 3;
              description "One or more references exist in storage.";
            }
            enum unknown {
              value 4;
              description "The location of the references are unknown.";
            }
          }
          description "Search scope for references in the operation.";
        }
        rpc configure {
            description "Configure RPC";
            input {
                uses client-id;
                uses execution-delay;
                uses ypatch:yang-patch;
            }
            output {
                uses ypatch:yang-patch-status;
            }
        }
        augment "/configure/input/yang-patch/edit" {
            leaf reference-scope {
                type fpc:ref-scope;
                description "Reference Scope";
            }
            uses fpcbase:instructions;
            description "yang-patch edit augments for configure rpc";
        }
        grouping subsequent-edits {
             list subsequent-edit {
               key edit-id;
               ordered-by user;

               description "Edit list";
```

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

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

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

            leaf point {
             when "(../operation = 'insert' or ../operation = 'move')"
              + "and (../where = 'before' or ../where = 'after')" {
                description
                  "This leaf only applies for 'insert' or 'move'
                   operations, before or after an existing entry.";
```

```
      }
      type ypatch:target-resource-offset;
      description
        "The absolute URL path for the data node";
    }

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

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

```
                "Value used for this edit operation.";
              }
            }
            description "Subsequent Edits";
        }
        augment "/configure/output/yang-patch-status/edit-status/edit/"
            + "edit-status-choice/ok" {
            leaf notify-follows {
                type boolean;
                description "Notify Follows Indication";
            }
            uses fpc:subsequent-edits;
            description "Configure output augments";
        }

        grouping op-header {
            uses client-id;
            uses execution-delay;
            leaf operation-id {
                type uint64;
                mandatory true;
                description "Operation Identifier";
            }
            description "Common Operation header";
        }
        grouping monitor-response {
            leaf operation-id {
                type uint64;
                mandatory true;
                description "Operation Identifier";
            }
            choice edit-status-choice {
                   description
                     "A choice between different types of status
                      responses for each 'edit' entry.";
                   leaf ok {
                     type empty;
                     description
                       "This 'edit' entry was invoked without any
                        errors detected by the server associated
                        with this edit.";
                   }
                   case errors {
                     uses rc:errors;
                     description
                       "The server detected errors associated with the
                        edit identified by the same 'edit-id' value.";
                   }
```

```
            }
            description "Monitor Response";
        }

      // Common RPCs
      rpc register_monitor {
        description "Used to register monitoring of parameters/events";
        input {
          uses fpc:op-header;
          list monitor {
            key monitor-key;
            uses fpc:monitor-config;
            description "Monitor Configuration";
          }
        }
        output {
            uses fpc:monitor-response;
        }
      }
      rpc deregister_monitor {
        description "Used to de-register monitoring of
          parameters/events";
        input {
          uses fpc:op-header;
          list monitor {
            key monitor-key;
            uses fpc:monitor-key;
            min-elements 1;
            leaf send_data {
                type boolean;
                description "Indicates if NOTIFY with final data
                    is desired upon deregistration";
            }
            description "Monitor Identifier";
          }
        }
        output {
            uses fpc:monitor-response;
        }
      }
      rpc probe {
        description "Probe the status of a registered monitor";
        input {
          uses fpc:op-header;
          list monitor {
            key monitor-key;
            uses fpc:monitor-key;
            min-elements 1;
```

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

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

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

```
    }
    identity monitoring-suspension {
      base "notification-cause";
      description "Indicates monitoring suspension";
    }
    identity monitoring-resumption {
      base "notification-cause";
      description "Indicates that monitoring has resumed";
    }
    identity dpn-available {
      base "notification-cause";
      description "DPN Candidate Available";
    }
    identity dpn-unavailable {
      base "notification-cause";
      description "DPN Unavailable";
    }
    notification notify {
      leaf notification-id {
        type uint32;
        description "Notification Identifier";
      }
      leaf timestamp {
        type uint32;
        description "timestamp";
      }
      list report {
          key monitor-key;
          uses fpc:monitor-key;
          min-elements 1;
          leaf trigger {
              type identityref {
                  base "notification-cause";
              }
              description "Notification Cause";
          }
          choice value {
              case dpn-candidate-available {
                  leaf node-id {
                      type inet:uri;
                      description "Topology URI";
                  }
                  list supported-interface-list {
                      key role-key;
                      uses fpc:role-key;
                      description "Support Intefaces";
                  }
                  description "DPN Candidate Information";
```

```
                }
                case dpn-unavailable {
                    leaf dpn-id {
                      type fpc:fpc-identity;
                      description "DPN Identifier for DPN Unavailable";
                    }
                    description "DPN Unavailable";
                }
                anydata report-value {
                    description "Any non integer report";
                }
                description "Report Value";
            }
            description "Report";
        }
        description "Notify Message";
      }
   }
   <CODE ENDS>
```

A.2. FPC YANG Settings and Extensions Model

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

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

```
<CODE BEGINS> file "ietf-dmm-fpc-settingsext@2018-05-17.yang"
module ietf-dmm-fpc-settingsext {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-dmm-fpc-settingsext";
  prefix fpcbase;

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

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

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

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

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

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

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

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

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

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

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

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

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

```
        }
        identity grev2 {
            base "fpcbase:tunnel-type";
            description "GRE v2";
        }
        identity ipinip {
            base "fpcbase:tunnel-type";
            description "IP in IP";
        }
        identity gtpv1 {
            base "fpcbase:tunnel-type";
            description "GTP version 1 Tunnel";
        }
        identity gtpv2 {
            base "fpcbase:tunnel-type";
            description "GTP version 2 Tunnel";
        }

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

```
                    enum dual {
                        value 2;
                        description "IPv4 and IPv6";
                    }
                }
                description "Payload Type";
            }
            leaf gre-key {
                type uint32;
                description "GRE_KEY";
            }
            container gtp-tunnel-info {
                leaf local-tunnel-identifier {
                    type uint32;
                    description "Tunnel Endpoint IDentifier (TEID)";
                }
                leaf remote-tunnel-identifier {
                    type uint32;
                    description "Tunnel Endpoint IDentifier (TEID)";
                }
                leaf sequence-numbers-enabled {
                    type boolean;
                    description "Sequence No. Enabled";
                }
                description "GTP Tunnel Information";
            }
            leaf ebi {
                type fpcbase:ebi-type;
                description "EPS Bearier Identifier";
            }
            leaf lbi {
                type fpcbase:ebi-type;
                description "Linked Bearier Identifier";
            }
            description "Tunnel Information";
          }
          description "Tunnel Value";
        }

      ////////////////////////////
      // DESCRIPTOR DEFINITIONS

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

```
            }
            enum uplink {
              value 1;
              description "uplink";
            }
            enum downlink {
              value 2;
              description "downlink";
            }
            enum bidirectional {
              value 3;
              description "bi-direcitonal";
            }
          }
          description "Packet Filter Direction";
        }
        typedef component-type-id {
          type uint8 {
            range "16 | 17 | 32 | 33 | 35 | 48 | 64 | 65 |"
            + " 80 | 81 | 96 | 112 | 128";
          }
          description "Specifies the Component Type";
        }
        grouping packet-filter {
          leaf direction {
            type fpcbase:packet-filter-direction;
            description "Filter Direction";
          }
          leaf identifier {
            type uint8 {
              range "1..15";
            }
            description "Filter Identifier";
          }
          leaf evaluation-precedence {
            type uint8;
            description "Evaluation Precedence";
          }
          list contents {
            key component-type-identifier;
            description "Filter Contents";
            leaf component-type-identifier {
              type fpcbase:component-type-id;
              description "Component Type";
            }
            choice value {
              leaf ipv4-local {
                type inet:ipv4-address;
```

```
                description "IPv4 Local Address";
            }
            leaf ipv6-prefix-local {
                type inet:ipv6-prefix;
                description "IPv6 Local Prefix";
            }
            leaf ipv4-ipv6-remote {
                type inet:ip-address;
                description "Ipv4 Ipv6 remote address";
            }
            leaf ipv6-prefix-remote {
                type inet:ipv6-prefix;
                description "IPv6 Remote Prefix";
            }
            leaf next-header {
                type uint8;
                description "Next Header";
            }
            leaf local-port {
                type inet:port-number;
                description "Local Port";
            }
            case local-port-range {
              leaf local-port-lo {
                type inet:port-number;
                description "Local Port Min Value";
              }
              leaf local-port-hi {
                type inet:port-number;
                description "Local Port Max Value";
              }
            }
            leaf remote-port {
                type inet:port-number;
                description "Remote Port";
            }
            case remote-port-range {
              leaf remote-port-lo {
                type inet:port-number;
                description "Remote Por Min Value";
              }
              leaf remote-port-hi {
                type inet:port-number;
                description "Remote Port Max Value";
              }
            }
            leaf ipsec-index {
                type traffic-selectors:ipsec-spi;
```

```
                description "IPSec Index";
            }
            leaf traffic-class {
                type inet:dscp;
                description "Traffic Class";
            }
            case traffic-class-range {
                leaf traffic-class-lo {
                  type inet:dscp;
                  description "Traffic Class Min Value";
                }
                leaf traffic-class-hi {
                  type inet:dscp;
                  description "Traffic Class Max Value";
                }
            }
            leaf-list flow-label {
                type inet:ipv6-flow-label;
                description "Flow Label";
            }
            description "Component Value";
          }
        }
        description "Packet Filter";
      }

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

  grouping fpc-descriptor-value {
    choice descriptor-value {
      mandatory true;
      leaf all-traffic {
          type empty;
          description "admit any";
      }
      leaf no-traffic {
          type empty;
```

```
            description "deny any";
          }
          case prefix-descriptor {
            uses fpcbase:prefix-descriptor;
            description "IP Prefix descriptor";
          }
          case pmip-selector {
            uses traffic-selectors:traffic-selector;
            description "PMIP Selector";
          }
          container rfc5777-classifier-template {
                uses rfc5777:classifier;
                description "RFC 5777 Classifier";
          }
          container packet-filter {
                uses fpcbase:packet-filter;
                description "Packet Filter";
          }
          case tunnel-info {
            uses fpcbase:tunnel-value;
            description "Tunnel Descriptor (only
                considers source info)";
          }
          description "Descriptor Value";
        }
        description "FPC Descriptor Values";
      }

        // Next Hop Structures
        typedef fpc-service-path-id {
            type uint32 {
                range "0..33554431";
            }
            description "SERVICE_PATH_ID";
        }
        typedef fpc-mpls-label {
            type uint32 {
              range "0..1048575";
            }
            description "MPLS label";
        }
        typedef segment-id {
            type string {
                length "16";
            }
            description "SR Segement Identifier";
        }
        grouping fpc-nexthop {
```

```
            choice next-hop-value {
                leaf ip-address {
                    type inet:ip-address;
                    description "IP Value";
                }
                leaf mac-address {
                    type ytypes:mac-address;
                    description "MAC Address Value";
                }
                leaf service-path {
                    type fpcbase:fpc-service-path-id;
                    description "Service Path Value";
                }
                leaf mpls-path {
                    type fpcbase:fpc-mpls-label;
                    description "MPLS Value";
                }
                leaf nsh {
                    type string {
                        length "16";
                    }
                    description "Network Service Header";
                }
                leaf interface {
                    type uint16;
                    description "If (interface) Value";
                }
                leaf segment-identifier {
                    type fpcbase:segment-id;
                    description "Segment Id";
                }
                leaf-list mpls-label-stack {
                    type fpcbase:fpc-mpls-label;
                    description "MPLS Stack";
                }
                leaf-list mpls-sr-stack {
                    type fpcbase:fpc-mpls-label;
                    description "MPLS SR Stack";
                }
                leaf-list srv6-stack {
                    type fpcbase:segment-id;
                    description "Segment Id";
                }
                case tunnel-info {
                    uses fpcbase:tunnel-value;
                    description "Tunnel Descriptor (only
                    considers source info)";
                }
```

```
            description "Value";
          }
          description "Nexthop Value";
        }

        ////////////////////////////
        // PMIP Integration         //
          typedef pmip-commandset {
                type bits {
                    bit assign-ip {
                      position 0;
                      description "Assign IP";
                    }
                    bit assign-dpn {
                      position 1;
                      description "Assign DPN";
                    }
                    bit session {
                      position 2;
                      description "Session Level";
                    }
                    bit uplink {
                      position 3;
                      description "Uplink";
                    }
                    bit downlink {
                      position 4;
                      description "Downlink";
                    }
                }
                description "PMIP Instructions";
          }
        ////////////////////////////
        // 3GPP Integration         //

          // Type Defs
          typedef fpc-qos-class-identifier {
              type uint8 {
                  range "1..9";
              }
              description "QoS Class Identifier (QCI)";
          }
          typedef ebi-type {
            type uint8 {
              range "0..15";
            }
            description "EUTRAN Bearere Identifier (EBI) Type";
          }
```

```
        typedef imsi-type {
            type uint64;
            description
                "International Mobile Subscriber Identity (IMSI)
                  Value Type";
        }
        // Instructions
        typedef threegpp-instr {
          type bits {
            bit assign-ip {
              position 0;
              description "Assign IP Address/Prefix";
            }
            bit assign-fteid-ip {
              position 1;
              description "Assign FTEID-IP";
            }
            bit assign-fteid-teid {
              position 2;
              description "Assign FTEID-TEID";
            }
            bit session {
              position 3;
              description "Commands apply to the Session Level";
            }
            bit uplink {
              position 4;
              description "Commands apply to the Uplink";
            }
            bit downlink {
              position 5;
              description "Commands apply to the Downlink";
            }
            bit assign-dpn {
              position 6;
              description "Assign DPN";
            }
          }
          description "Instruction Set for 3GPP R11";
        }

       ////////////////////////////
       // ACTION VALUE AUGMENTS
       grouping fpc-action-value {
           choice action-value {
               mandatory true;
               leaf drop {
                   type empty;
```

```
                    description "Drop Traffic";
                }
                container rewrite {
                    choice rewrite-value {
                        case prefix-descriptor {
                            uses fpcbase:prefix-descriptor;
                            description "IP Prefix descriptor";
                        }
                        case pmip-selector {
                            uses traffic-selectors:traffic-selector;
                            description "PMIP Selector";
                        }
                        container rfc5777-classifier-template {
                            uses rfc5777:classifier;
                            description "RFC 5777 Classifier";
                        }
                        description "Rewrite Choice";
                    }
                    description "Rewrite/NAT value";
                }
                container copy-forward-nexthop {
                        uses fpcbase:fpc-nexthop;
                        description "Copy Forward Value";
                }
                container nexthop {
                        uses fpcbase:fpc-nexthop;
                        description "NextHop Value";
                }
                case qos {
                    leaf trafficclass {
                        type inet:dscp;
                        description "Traffic Class";
                    }
                    uses pmipqos:qosattribute;
                    leaf qci {
                        type fpcbase:fpc-qos-class-identifier;
                        description "QCI";
                    }
                    leaf ue-agg-max-bitrate {
                        type uint32;
                        description "UE Aggregate Max Bitrate";
                    }
                    leaf apn-ambr {
                        type uint32;
                        description
                         "Access Point Name Aggregate Max Bit Rate";
                    }
                    description "QoS Attributes";
```

```
                }
                description "Action Value";
            }
            description "FPC Action Value";
        }

      // Instructions
      grouping instructions {
        container command-set {
          choice instr-type {
            leaf instr-3gpp-mob {
                type fpcbase:threegpp-instr;
                description "3GPP GTP Mobility Instructions";
            }
            leaf instr-pmip {
                type pmip-commandset;
                description "PMIP Instructions";
            }
            description "Instruction Value Choice";
          }
          description "Instructions";
        }
        description "Instructions Value";
      }
    }
    <CODE ENDS>
```

A.3. PMIP QoS Model

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

This module references [RFC6991].

```
<CODE BEGINS> file "ietf-pmip-qos@2018-05-17.yang"
module ietf-pmip-qos {
    yang-version 1.1;

    namespace
      "urn:ietf:params:xml:ns:yang:ietf-pmip-qos";

    prefix "qos-pmip";

    import ietf-inet-types {
      prefix inet;
      revision-date 2013-07-15;
    }
    import ietf-trafficselector-types { prefix traffic-selectors;
```

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

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

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

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

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

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

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

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

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

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

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

       // Type Definitions

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

```
          description
           "An 8-bit unsigned integer used for identifying the QoS
            Service Request.";
        }

        typedef traffic-class {
          type inet:dscp;
          description
            "Traffic Class consists of a 6-bit DSCP field followed by a
             2-bit reserved field.";
         reference
             "RFC 3289: Management Information Base for the
                Differentiated Services Architecture
              RFC 2474: Definition of the Differentiated Services Field
                        (DS Field) in the IPv4 and IPv6 Headers
              RFC 2780: IANA Allocation Guidelines For Values In
                        the Internet Protocol and Related Headers";
        }

        typedef operational-code {
          type enumeration {
            enum RESPONSE {
              value 0;
              description "Response to a QoS request";
            }
            enum ALLOCATE {
              value 1;
              description "Request to allocate QoS resources";
            }
            enum DE-ALLOCATE {
              value 2;
              description "Request to de-Allocate QoS resources";
            }
            enum MODIFY {
              value 3;
              description "Request to modify QoS parameters for a
                  previously negotiated QoS Service Request";
            }
            enum QUERY {
              value 4;
              description "Query to list the previously negotiated QoS
                  Service Requests that are still active";
            }
            enum NEGOTIATE {
              value 5;
              description "Response to a QoS Service Request with a
                counter QoS proposal";
            }
```

```
        }
        description
         "The type of QoS request. Reserved values:   (6) to (255)
                  Currently not used.  Receiver MUST ignore the option
                  received with any value in this range.";
       }

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

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

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

```
          description
           "This flag is used to request that the uplink/downlink
          flows for which the network is providing
               Guaranteed-Bit-Rate service be excluded from the
               target IP flows for which
               Per-Session-Agg-Max-UL/DL-Bit-Rate is measured.";
        }
       description "Per-Session-Agg-Max-Bit-Rate Value";
      }

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

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

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

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

       typedef Guaranteed-DL-Bit-Rate-Value {
         type uint32;
```

```
        description
        "The guaranteed bandwidth in bits per second for downlink
          IP flows.  The measurement units are bits per second.";
      }

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

      grouping QoS-Vendor-Specific-Attribute-Value-Base {
        leaf vendorid {
          type uint32;
          mandatory true;
          description
           "The Vendor ID is the SMI (Structure of Management
            Information) Network Management Private Enterprise Code of
            the IANA-maintained 'Private Enterprise Numbers'
            registry.";
          reference
            "'PRIVATE ENTERPRISE NUMBERS', SMI Network Management
              Private Enterprise Codes, April 2014,
               <http://www.iana.org/assignments/enterprise-numbers>";
        }
        leaf subtype {
          type uint8;
          mandatory true;
          description
            "An 8-bit field indicating the type of vendor-specific
             information carried in the option.  The namespace for this
             sub-type is managed by the vendor identified by the
             Vendor ID field.";
        }
        description
          "QoS Vendor-Specific Attribute.";
      }

      //Primary Structures (groupings)
      grouping qosattribute {
          leaf per-mn-agg-max-dl {
              type qos-pmip:Per-MN-Agg-Max-DL-Bit-Rate-Value;
              description "Per-MN-Agg-Max-DL-Bit-Rate Value";
          }
          leaf per-mn-agg-max-ul {
              type qos-pmip:Per-MN-Agg-Max-UL-Bit-Rate-Value;
              description "Per-MN-Agg-Max-UL-Bit-Rate Value";
```

```
            }
            container per-session-agg-max-dl {
                uses qos-pmip:Per-Session-Agg-Max-Bit-Rate-Value;
                description "Per-Session-Agg-Max-Bit-Rate Value";
            }
            container per-session-agg-max-ul {
                uses qos-pmip:Per-Session-Agg-Max-Bit-Rate-Value;
                description "Per-Session-Agg-Max-Bit-Rate Value";
            }
            uses qos-pmip:Allocation-Retention-Priority-Value;
            leaf agg-max-dl {
                type qos-pmip:Aggregate-Max-DL-Bit-Rate-Value;
                description "Aggregate-Max-DL-Bit-Rate Value";
            }
            leaf agg-max-ul {
                type qos-pmip:Aggregate-Max-UL-Bit-Rate-Value;
                description "Aggregate-Max-UL-Bit-Rate Value";
            }
            leaf gbr-dl {
                type qos-pmip:Guaranteed-DL-Bit-Rate-Value;
                description "Guaranteed-DL-Bit-Rate Value";
            }
            leaf gbr-ul {
                type qos-pmip:Guaranteed-UL-Bit-Rate-Value;
                description "Guaranteed-UL-Bit-Rate Value";
            }
            description "PMIP QoS Attributes. Note Vendor option
            is not a part of this grouping";
        }

        grouping qosoption {
            leaf srid {
                type sr-id;
                mandatory true;
                description "Service Request Identifier";
            }
            leaf trafficclass {
                type traffic-class;
                mandatory true;
                description "Traffic Class";
            }
            leaf operationcode {
                type operational-code;
                mandatory true;
                description "Operation Code";
            }
            uses qos-pmip:qosattribute;
            uses qos-pmip:QoS-Vendor-Specific-Attribute-Value-Base;
```

```
            container traffic-selector {
                uses traffic-selectors:traffic-selector;
                description "traffic selector";
            }
            description "PMIP QoS Option";
        }
    }
    <CODE ENDS>
```

A.4. Traffic Selectors YANG Model

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

This module references [RFC6991].

```
<CODE BEGINS> file "ietf-trafficselector-types@2018-05-17.yang"
module ietf-trafficselector-types {
 yang-version 1.1;

 namespace
 "urn:ietf:params:xml:ns:yang:ietf-trafficselector-types";

 prefix "traffic-selectors";

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

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

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

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

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

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

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

```
    description
    "This module contains a collection of YANG definitions for
    traffic selectors for flow bindings.

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

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

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

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

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

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

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

```
          reference
          "RFC 4303: IP Encapsulating Security
          Payload (ESP)";
        }

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

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

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

```
            "The last source address to be matched";
          }
        }
        container destination-address-range-v4 {
           presence "Enables setting destination IPv4 address range";
           description
             "Inclusive range representing IPv4 addresses to be used.
             When only start-address is present, it represents a
             single address.";
           leaf start-address {
             type inet:ipv4-address;
             mandatory true;
             description
              "The first destination address to be matched";
           }
           leaf end-address {
             type inet:ipv4-address;
             description
              "The last destination address to be matched";
           }
        }
        container ds-range {
           presence "Enables setting dscp range";
           description
            "Inclusive range representing DiffServ Codepoints to be used.
            When only start-ds is present, it represents a single
            Codepoint.";
           leaf start-ds {
             type inet:dscp;
             mandatory true;
             description
              "The first differential service value to be matched";
         }
         leaf end-ds {
           type inet:dscp;
           must ". >= ../start-ds" {
             error-message
               "The end-ds must be greater than or equal to start-ds";
           }
           description
             "The last differential service value to be matched";
         }
        }
        container protocol-range {
          presence "Enables setting protocol range";
          description
            "Inclusive range representing IP protocol(s) to be used. When
             only start-protocol is present, it represents a single
```

```
          protocol.";
        leaf start-protocol {
          type uint8;
          mandatory true;
          description
            "The first 8-bit protocol value to be matched.";
        }
        leaf end-protocol {
          type uint8;
          must ". >= ../start-protocol" {
            error-message
              "The end-protocol must be greater than or equal to
             start-protocol";
          }
        description
          "The last 8-bit protocol value to be matched.";
        }
      }
      description "ipv4 binary traffic selector";
    }
     grouping ipv6-binary-traffic-selector {
      container source-address-range-v6 {
        presence "Enables setting source IPv6 address range";
         description
          "Inclusive range representing IPv6 addresses to be used.
          When only start-address is present, it represents a
          single address.";
         leaf start-address {
           type inet:ipv6-address;
           mandatory true;
           description
           "The first source address, from the
           range of 128-bit IPv6 addresses to be matched";
         }
         leaf end-address {
           type inet:ipv6-address;
           description
               "The last source address, from the
               range of 128-bit IPv6 addresses to be matched";
         }
      }
      container destination-address-range-v6 {
        presence "Enables setting destination IPv6 address range";
        description
          "Inclusive range representing IPv6 addresses to be used.
           When only start-address is present, it represents a
           single address.";
        leaf start-address {
```

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

```
           (ECN) to IP";
       }
       leaf end-traffic-class {
         type inet:dscp;
         must ". >= ../start-traffic-class" {
           error-message
             "The end-traffic-class must be greater than or equal to
              start-traffic-class";
         }
         description
           "The last traffic class value to be matched";
       }
     }
     container next-header-range {
       presence "Enables setting Next Header range";
       description
        "Inclusive range representing Next Headers to be used. When
         only start-next-header is present, it represents a
         single Next Header.";
       leaf start-next-header {
         type uint8;
         description
          "The first 8-bit next header value to be matched.";
       }
       leaf end-next-header {
         type uint8;
         must ". >= ../start-next-header" {
           error-message
             "The end-next-header must be greater than or equal to
              start-next-header";
         }
         description
           "The last 8-bit next header value to be matched.";
       }
     }
     description "ipv6 binary traffic selector";
   }

     grouping traffic-selector {
       leaf ts-format {
          type identityref {
            base traffic-selector-format;
          }
          description "Traffic Selector Format";
       }
       uses traffic-selectors:traffic-selector-base;
       uses traffic-selectors:ipv4-binary-traffic-selector;
       uses traffic-selectors:ipv6-binary-traffic-selector;
```

```
       description
        "The traffic selector includes the parameters used to match
          packets for a specific flow binding.";
       reference
        "RFC 6089: Flow Bindings in Mobile IPv6 and Network
          Mobility (NEMO) Basic Support";
     }
   }
   <CODE ENDS>
```

A.5. RFC 5777 Classifier YANG Model

This module defines the RFC 5777 Classifer.

This module references [RFC5777].

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

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

 prefix "diamclassifier";

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

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

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

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

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

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

 Editor: Lyle Bertz
```

```
    <mailto:lylebe551144@gmail.com>";

    description
    "This module contains a collection of YANG definitions for
    traffic classification and QoS Attributes for Diameter.

    Copyright (c) 2018 IETF Trust and the persons identified as the
    document authors. All rights reserved.

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

     revision 2018-05-17 {
       description
           "Initial";
       reference
       "RFC 5777: Traffic Classification and Quality of Service (QoS)
           Attributes for Diameter";
     }

       typedef eui64-address-type {
           type string {
               length "6";
           }
           description
               "specifies a single layer 2 address in EUI-64 format.
                The value is an 8-octet encoding of the address as
                it would appear in the frame header.";
       }
       typedef direction-type {
           type enumeration {
               enum IN {
                   value 0;
                   description
                       "Applies to flows from the managed terminal.";
               }
               enum OUT {
                   value 1;
                   description
                    "Applies to flows to the managed terminal.";
               }
```

```
                enum BOTH {
                    value 2;
                    description
                        "Applies to flows both to and from the managed
                            terminal.";
                }
            }
            description
                "Specifies in which direction to apply the classifier.";
        }
        typedef negated-flag-type {
            type enumeration {
                enum False { value 0;
                    description "false"; }
                enum True { value 1;
                    description "True"; }
            }
            description
                "When set to True, the meaning of the match is
                 inverted and the classifier will match addresses
                 other than those specified by the From-Spec or
                 To-Spec AVP.

                 Note that the negation does not impact the port
                    comparisons.";
        }
        grouping index {
            leaf index {
                type uint16;
                mandatory true;
                description "Identifier used for referencing";
            }
            description "Index Value";
        }
        grouping to-from-spec-value {
            leaf-list ip-address {
                type inet:ip-address;
                description "IP address";
            }
            list ip-address-range {
                key index;
                uses diamclassifier:index;
                leaf ip-address-start {
                    type inet:ip-address;
                    description "IP Address Start";
                }
                leaf ip-address-end {
                    type inet:ip-address;
```

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

```
            }
            leaf-list port {
                type inet:port-number;
                description "Port Number";
            }
            list port-range {
                key index;
                uses diamclassifier:index;
                leaf ip-address-start {
                    type inet:port-number;
                    description "Port Start";
                }
                leaf ip-address-end {
                    type inet:port-number;
                    description "Port End";
                }
                description "Port Range";
            }
            leaf negated {
                type diamclassifier:negated-flag-type;
                description "Negated";
            }
            leaf use-assigned-address {
                type boolean;
                description "Use Assigned Address";
            }
            description
                "Basic traffic description value";
        }

        grouping option-type-group {
            leaf option-type {
                type uint8;
                mandatory true;
                description "Option Type";
            }
            leaf-list ip-option-value {
                type string;
                description "Option Value";
            }
            leaf negated {
                type diamclassifier:negated-flag-type;
                description "Negated";
            }
            description "Common X Option Pattern";
        }
        typedef vlan-id {
            type uint32 {
```

```
                range "0..4095";
            }
            description "VLAN ID";
        }

      grouping classifier {
        leaf protocol {
            type uint8;
            description "Protocol";
        }
        leaf direction {
            type diamclassifier:direction-type;
            description "Direction";
        }
        list from-spec {
            key index;
            uses diamclassifier:index;
            uses diamclassifier:to-from-spec-value;
            description "from specification";
        }
        list to-spec {
            key index;
            uses diamclassifier:index;
            uses diamclassifier:to-from-spec-value;
            description "to specification";
        }
        leaf-list disffserv-code-point {
            type inet:dscp;
            description "DSCP";
        }
        leaf fragmentation-flag {
        type enumeration {
                enum DF {
                    value 0;
                    description "Don’t Fragment";
                }
                enum MF {
                    value 1;
                    description "More Fragments";
                }
            }
            description "Fragmenttation Flag";
        }
        list ip-option {
            key option-type;
            uses diamclassifier:option-type-group;
            description "IP Option Value";
        }
```

```
        list tcp-option {
            key option-type;
            uses diamclassifier:option-type-group;
            description "TCP Option Value";
        }
        list tcp-flag {
            key tcp-flag-type;
            leaf tcp-flag-type {
                type uint32;
                mandatory true;
                description "TCP Flag Type";
            }
            leaf negated {
                type diamclassifier:negated-flag-type;
                description "Negated";
            }
            description "TCP Flags";
        }
        list icmp-option {
            key option-type;
            uses diamclassifier:option-type-group;
            description "ICMP Option Value";
        }
        list eth-option {
            key index;
            uses diamclassifier:index;
            container eth-proto-type {
                leaf-list eth-ether-type {
                    type string {
                        length "2";
                    }
                    description "value of ethertype field";
                }
                leaf-list eth-sap {
                    type string {
                        length "2";
                    }
                    description "802.2 SAP";
                }
                description "Ether Proto Type";
            }
            list vlan-id-range {
                key index;
                uses diamclassifier:index;
                leaf-list s-vlan-id-start {
                    type diamclassifier:vlan-id;
                    description "S-VID  VLAN ID Start";
                }
```

```
                leaf-list s-vlan-id-end {
                    type diamclassifier:vlan-id;
                    description "S-VID VLAN ID End";
                }
                leaf-list c-vlan-id-start {
                    type diamclassifier:vlan-id;
                    description "C-VID  VLAN ID Start";
                }
                leaf-list c-vlan-id-end {
                    type diamclassifier:vlan-id;
                    description "C-VID  VLAN ID End";
                }
                description "VLAN ID Range";
            }
            list user-priority-range {
                key index;
                uses diamclassifier:index;
                leaf-list low-user-priority {
                    type uint32 {
                        range "0..7";
                    }
                    description "Low User Priority";
                }
                leaf-list high-user-priority {
                    type uint32 {
                        range "0..7";
                    }
                    description "High User Priority";
                }
                description "User priority range";
            }
            description "Ether Option";
        }
        description "RFC 5777 Classifier";
    }
}
<CODE ENDS>
```

Appendix B. FPC YANG Tree Structure

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

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

```
     |  +--rw no-traffic?                    empty
     +--:(prefix-descriptor)
     |  +--rw destination-ip?                inet:ip-prefix
     |  +--rw source-ip?                     inet:ip-prefix
     +--:(pmip-selector)
     |  +--rw ts-format?                     identityref
     |  +--rw ipsec-spi-range!
     |  |  +--rw start-spi    ipsec-spi
     |  |  +--rw end-spi?     ipsec-spi
     |  +--rw source-port-range!
     |  |  +--rw start-port    inet:port-number
     |  |  +--rw end-port?     inet:port-number
     |  +--rw destination-port-range!
     |  |  +--rw start-port    inet:port-number
     |  |  +--rw end-port?     inet:port-number
     |  +--rw source-address-range-v4!
     |  |  +--rw start-address    inet:ipv4-address
     |  |  +--rw end-address?     inet:ipv4-address
     |  +--rw destination-address-range-v4!
     |  |  +--rw start-address    inet:ipv4-address
     |  |  +--rw end-address?     inet:ipv4-address
     |  +--rw ds-range!
     |  |  +--rw start-ds    inet:dscp
     |  |  +--rw end-ds?     inet:dscp
     |  +--rw protocol-range!
     |  |  +--rw start-protocol    uint8
     |  |  +--rw end-protocol?     uint8
     |  +--rw source-address-range-v6!
     |  |  +--rw start-address    inet:ipv6-address
     |  |  +--rw end-address?     inet:ipv6-address
     |  +--rw destination-address-range-v6!
     |  |  +--rw start-address    inet:ipv6-address
     |  |  +--rw end-address?     inet:ipv6-address
     |  +--rw flow-label-range!
     |  |  +--rw start-flow-label?   inet:ipv6-flow-label
     |  |  +--rw end-flow-label?     inet:ipv6-flow-label
     |  +--rw traffic-class-range!
     |  |  +--rw start-traffic-class?   inet:dscp
     |  |  +--rw end-traffic-class?     inet:dscp
     |  +--rw next-header-range!
     |     +--rw start-next-header?   uint8
     |     +--rw end-next-header?     uint8
     +--:(rfc5777-classifier-template)
     |  +--rw rfc5777-classifier-template
     |     +--rw protocol?               uint8
     |     +--rw direction?              diamclassifier:direction-type
     |     +--rw from-spec* [index]
     |     |  +--rw index                   uint16
```

```
|     |  +--rw ip-address*             inet:ip-address
|     |  +--rw ip-address-range* [index]
|     |  |  +--rw index                uint16
|     |  |  +--rw ip-address-start?    inet:ip-address
|     |  |  +--rw ip-address-end?      inet:ip-address
|     |  +--rw ip-address-mask*         inet:ip-prefix
|     |  +--rw mac-address*             yang-types:mac-address
|     |  +--rw mac-address-mask* [mac-address]
|     |  |  +--rw mac-address              yang-types:mac-address
|     |  |  +--rw macaddress-mask-pattern  yang-types:mac-address
|     |  +--rw eui64-address*
                     diamclassifier:eui64-address-type
|     |  +--rw eui64-address-mask* [eui64-address]
|     |  |  +--rw eui64-address
                     diamclassifier:eui64-address-type
|     |  |  +--rw eui64-address-mask-pattern
                     diamclassifier:eui64-address-type
|     |  +--rw port*                    inet:port-number
|     |  +--rw port-range* [index]
|     |  |  +--rw index                uint16
|     |  |  +--rw ip-address-start?    inet:port-number
|     |  |  +--rw ip-address-end?      inet:port-number
|     |  +--rw negated?
                     diamclassifier:negated-flag-type
|     |  +--rw use-assigned-address?   boolean
|     +--rw to-spec* [index]
|     |  +--rw index                    uint16
|     |  +--rw ip-address*              inet:ip-address
|     |  +--rw ip-address-range* [index]
|     |  |  +--rw index                uint16
|     |  |  +--rw ip-address-start?    inet:ip-address
|     |  |  +--rw ip-address-end?      inet:ip-address
|     |  +--rw ip-address-mask*         inet:ip-prefix
|     |  +--rw mac-address*             yang-types:mac-address
|     |  +--rw mac-address-mask* [mac-address]
|     |  |  +--rw mac-address                yang-types:mac-address
|     |  |  +--rw macaddress-mask-pattern    yang-types:mac-address
|     |  +--rw eui64-address*
                     diamclassifier:eui64-address-type
|     |  +--rw eui64-address-mask* [eui64-address]
|     |  |  +--rw eui64-address
                     diamclassifier:eui64-address-type
|     |  |  +--rw eui64-address-mask-pattern
                     diamclassifier:eui64-address-type
|     |  +--rw port*                    inet:port-number
|     |  +--rw port-range* [index]
|     |  |  +--rw index                uint16
|     |  |  +--rw ip-address-start?    inet:port-number
```

```
|     |  |  +--rw ip-address-end?     inet:port-number
|     |  +--rw negated?
                    diamclassifier:negated-flag-type
|     |  +--rw use-assigned-address?   boolean
|     +--rw disffserv-code-point*   inet:dscp
|     +--rw fragmentation-flag?     enumeration
|     +--rw ip-option* [option-type]
|     |  +--rw option-type        uint8
|     |  +--rw ip-option-value*   string
|     |  +--rw negated?           diamclassifier:negated-flag-type
|     +--rw tcp-option* [option-type]
|     |  +--rw option-type        uint8
|     |  +--rw ip-option-value*   string
|     |  +--rw negated?           diamclassifier:negated-flag-type
|     +--rw tcp-flag* [tcp-flag-type]
|     |  +--rw tcp-flag-type    uint32
|     |  +--rw negated?         diamclassifier:negated-flag-type
|     +--rw icmp-option* [option-type]
|     |  +--rw option-type        uint8
|     |  +--rw ip-option-value*   string
|     |  +--rw negated?           diamclassifier:negated-flag-type
|     +--rw eth-option* [index]
|        +--rw index                 uint16
|        +--rw eth-proto-type
|        |  +--rw eth-ether-type*   string
|        |  +--rw eth-sap*          string
|        +--rw vlan-id-range* [index]
|        |  +--rw index              uint16
|        |  +--rw s-vlan-id-start*   diamclassifier:vlan-id
|        |  +--rw s-vlan-id-end*     diamclassifier:vlan-id
|        |  +--rw c-vlan-id-start*   diamclassifier:vlan-id
|        |  +--rw c-vlan-id-end*     diamclassifier:vlan-id
|        +--rw user-priority-range* [index]
|           +--rw index                uint16
|           +--rw low-user-priority*    uint32
|           +--rw high-user-priority*   uint32
+--:(packet-filter)
|  +--rw packet-filter
|     +--rw direction?            fpcbase:packet-filter-direction
|     +--rw identifier?             uint8
|     +--rw evaluation-precedence?   uint8
|     +--rw contents* [component-type-identifier]
|        +--rw component-type-identifier fpcbase:component-type-id
|        +--rw (value)?
|           +--:(ipv4-local)
|           |  +--rw ipv4-local?                 inet:ipv4-address
|           +--:(ipv6-prefix-local)
|           |  +--rw ipv6-prefix-local?          inet:ipv6-prefix
```

```
     |           +--:(ipv4-ipv6-remote)
     |           |  +--rw ipv4-ipv6-remote?           inet:ip-address
     |           +--:(ipv6-prefix-remote)
     |           |  +--rw ipv6-prefix-remote?         inet:ipv6-prefix
     |           +--:(next-header)
     |           |  +--rw next-header?                uint8
     |           +--:(local-port)
     |           |  +--rw local-port?                 inet:port-number
     |           +--:(local-port-range)
     |           |  +--rw local-port-lo?              inet:port-number
     |           |  +--rw local-port-hi?              inet:port-number
     |           +--:(remote-port)
     |           |  +--rw remote-port?                inet:port-number
     |           +--:(remote-port-range)
     |           |  +--rw remote-port-lo?             inet:port-number
     |           |  +--rw remote-port-hi?             inet:port-number
     |           +--:(ipsec-index)
     |           |  +--rw ipsec-index?     traffic-selectors:ipsec-spi
     |           +--:(traffic-class)
     |           |  +--rw traffic-class?              inet:dscp
     |           +--:(traffic-class-range)
     |           |  +--rw traffic-class-lo?           inet:dscp
     |           |  +--rw traffic-class-hi?           inet:dscp
     |           +--:(flow-label)
     |              +--rw flow-label*    inet:ipv6-flow-label
     +--:(tunnel-info)
        +--rw tunnel-info
           +--rw tunnel-local-address?    inet:ip-address
           +--rw tunnel-remote-address?   inet:ip-address
           +--rw mtu-size?                uint32
           +--rw tunnel?                  identityref
           +--rw payload-type?            enumeration
           +--rw gre-key?                 uint32
           +--rw gtp-tunnel-info
           |  +--rw local-tunnel-identifier?    uint32
           |  +--rw remote-tunnel-identifier?   uint32
           |  +--rw sequence-numbers-enabled?   boolean
           +--rw ebi?                     fpcbase:ebi-type
           +--rw lbi?                     fpcbase:ebi-type

   action_value:
   +--:(action-value)
   |  +--rw (action-value)
   |     +--:(drop)
   |     |  +--rw drop?                          empty
   |     +--:(rewrite)
   |     |  +--rw rewrite
   |     |     +--rw (rewrite-value)?
```

```
   |     |        +--:(prefix-descriptor)
   |     |        |  +--rw destination-ip?              inet:ip-prefix
   |     |        |  +--rw source-ip?                   inet:ip-prefix
   |     |        +--:(pmip-selector)
   |     |        |  +--rw ts-format?                     identityref
   |     |        |  +--rw ipsec-spi-range!
   |     |        |  |  +--rw start-spi    ipsec-spi
   |     |        |  |  +--rw end-spi?     ipsec-spi
   |     |        |  +--rw source-port-range!
   |     |        |  |  +--rw start-port    inet:port-number
   |     |        |  |  +--rw end-port?     inet:port-number
   |     |        |  +--rw destination-port-range!
   |     |        |  |  +--rw start-port    inet:port-number
   |     |        |  |  +--rw end-port?     inet:port-number
   |     |        |  +--rw source-address-range-v4!
   |     |        |  |  +--rw start-address    inet:ipv4-address
   |     |        |  |  +--rw end-address?     inet:ipv4-address
   |     |        |  +--rw destination-address-range-v4!
   |     |        |  |  +--rw start-address    inet:ipv4-address
   |     |        |  |  +--rw end-address?     inet:ipv4-address
   |     |        |  +--rw ds-range!
   |     |        |  |  +--rw start-ds    inet:dscp
   |     |        |  |  +--rw end-ds?     inet:dscp
   |     |        |  +--rw protocol-range!
   |     |        |  |  +--rw start-protocol    uint8
   |     |        |  |  +--rw end-protocol?     uint8
   |     |        |  +--rw source-address-range-v6!
   |     |        |  |  +--rw start-address    inet:ipv6-address
   |     |        |  |  +--rw end-address?     inet:ipv6-address
   |     |        |  +--rw destination-address-range-v6!
   |     |        |  |  +--rw start-address    inet:ipv6-address
   |     |        |  |  +--rw end-address?     inet:ipv6-address
   |     |        |  +--rw flow-label-range!
   |     |        |  |  +--rw start-flow-label?   inet:ipv6-flow-label
   |     |        |  |  +--rw end-flow-label?     inet:ipv6-flow-label
   |     |        |  +--rw traffic-class-range!
   |     |        |  |  +--rw start-traffic-class?   inet:dscp
   |     |        |  |  +--rw end-traffic-class?     inet:dscp
   |     |        |  +--rw next-header-range!
   |     |        |     +--rw start-next-header?   uint8
   |     |        |     +--rw end-next-header?     uint8
   |     |        +--:(rfc5777-classifier-template)
   |     |           +--rw rfc5777-classifier-template
   |     |              +--rw protocol?               uint8
   |     |              +--rw direction?
                         diamclassifier:direction-type
   |     |              +--rw from-spec* [index]
   |     |              |  +--rw index                  uint16
```

```
   |     |              |  +--rw ip-address*             inet:ip-address
   |     |              |  +--rw ip-address-range* [index]
   |     |              |  |  +--rw index               uint16
   |     |              |  |  +--rw ip-address-start?   inet:ip-address
   |     |              |  |  +--rw ip-address-end?     inet:ip-address
   |     |              |  +--rw ip-address-mask*        inet:ip-prefix
   |     |              |  +--rw mac-address*    yang-types:mac-address
   |     |              |  +--rw mac-address-mask* [mac-address]
   |     |              |  |  +--rw mac-address
                                  yang-types:mac-address
   |     |              |  |  +--rw macaddress-mask-pattern
                                  yang-types:mac-address
   |     |              |  +--rw eui64-address*
                                  diamclassifier:eui64-address-type
   |     |              |  +--rw eui64-address-mask* [eui64-address]
   |     |              |  |  +--rw eui64-address
                                  diamclassifier:eui64-address-type
   |     |              |  |  +--rw eui64-address-mask-pattern
                                  diamclassifier:eui64-address-type
   |     |              |  +--rw port*                inet:port-number
   |     |              |  +--rw port-range* [index]
   |     |              |  |  +--rw index               uint16
   |     |              |  |  +--rw ip-address-start?   inet:port-number
   |     |              |  |  +--rw ip-address-end?     inet:port-number
   |     |              |  +--rw negated?
                                  diamclassifier:negated-flag-type
   |     |              |  +--rw use-assigned-address?   boolean
   |     |              +--rw to-spec* [index]
   |     |              |  +--rw index                   uint16
   |     |              |  +--rw ip-address*             inet:ip-address
   |     |              |  +--rw ip-address-range* [index]
   |     |              |  |  +--rw index               uint16
   |     |              |  |  +--rw ip-address-start?   inet:ip-address
   |     |              |  |  +--rw ip-address-end?     inet:ip-address
   |     |              |  +--rw ip-address-mask*        inet:ip-prefix
   |     |              |  +--rw mac-address*
                                  yang-types:mac-address
   |     |              |  +--rw mac-address-mask* [mac-address]
   |     |              |  |  +--rw mac-address
                                  yang-types:mac-address
   |     |              |  |  +--rw macaddress-mask-pattern
                                  yang-types:mac-address
   |     |              |  +--rw eui64-address*
                                  diamclassifier:eui64-address-type
   |     |              |  +--rw eui64-address-mask* [eui64-address]
   |     |              |  |  +--rw eui64-address
                                  diamclassifier:eui64-address-type
   |     |              |  |  +--rw eui64-address-mask-pattern
```

```
                                    diamclassifier:eui64-address-type
   |     |              |  +--rw port*                  inet:port-number
   |     |              |  +--rw port-range* [index]
   |     |              |  |  +--rw index                uint16
   |     |              |  |  +--rw ip-address-start?   inet:port-number
   |     |              |  |  +--rw ip-address-end?     inet:port-number
   |     |              |  +--rw negated?
                                    diamclassifier:negated-flag-type
   |     |              |  +--rw use-assigned-address?   boolean
   |     |              +--rw disffserv-code-point*   inet:dscp
   |     |              +--rw fragmentation-flag?     enumeration
   |     |              +--rw ip-option* [option-type]
   |     |              |  +--rw option-type        uint8
   |     |              |  +--rw ip-option-value*   string
   |     |              |  +--rw negated?
                                    diamclassifier:negated-flag-type
   |     |              +--rw tcp-option* [option-type]
   |     |              |  +--rw option-type        uint8
   |     |              |  +--rw ip-option-value*   string
   |     |              |  +--rw negated?
                                    diamclassifier:negated-flag-type
   |     |              +--rw tcp-flag* [tcp-flag-type]
   |     |              |  +--rw tcp-flag-type    uint32
   |     |              |  +--rw negated?
                                    diamclassifier:negated-flag-type
   |     |              +--rw icmp-option* [option-type]
   |     |              |  +--rw option-type        uint8
   |     |              |  +--rw ip-option-value*   string
   |     |              |  +--rw negated?
                                    diamclassifier:negated-flag-type
   |     |              +--rw eth-option* [index]
   |     |                 +--rw index                  uint16
   |     |                 +--rw eth-proto-type
   |     |                 |  +--rw eth-ether-type*   string
   |     |                 |  +--rw eth-sap*          string
   |     |                 +--rw vlan-id-range* [index]
   |     |                 |  +--rw index             uint16
   |     |                 |  +--rw s-vlan-id-start*
                                    diamclassifier:vlan-id
   |     |                 |  +--rw s-vlan-id-end*
                                    diamclassifier:vlan-id
   |     |                 |  +--rw c-vlan-id-start*
                                    diamclassifier:vlan-id
   |     |                 |  +--rw c-vlan-id-end*
                                    diamclassifier:vlan-id
   |     |                 +--rw user-priority-range* [index]
   |     |                    +--rw index                uint16
   |     |                    +--rw low-user-priority*   uint32
```

```
|     |                    +--rw high-user-priority*   uint32
|     +--:(copy-forward-nexthop)
|     |  +--rw copy-forward-nexthop
|     |     +--rw (next-hop-value)?
|     |        +--:(ip-address)
|     |        |  +--rw ip-address?           inet:ip-address
|     |        +--:(mac-address)
|     |        |  +--rw mac-address?          ytypes:mac-address
|     |        +--:(service-path)
|     |        |  +--rw service-path?   fpcbase:fpc-service-path-id
|     |        +--:(mpls-path)
|     |        |  +--rw mpls-path?            fpcbase:fpc-mpls-label
|     |        +--:(nsh)
|     |        |  +--rw nsh?                  string
|     |        +--:(interface)
|     |        |  +--rw interface?            uint16
|     |        +--:(segment-identifier)
|     |        |  +--rw segment-identifier?   fpcbase:segment-id
|     |        +--:(mpls-label-stack)
|     |        |  +--rw mpls-label-stack*     fpcbase:fpc-mpls-label
|     |        +--:(mpls-sr-stack)
|     |        |  +--rw mpls-sr-stack*        fpcbase:fpc-mpls-label
|     |        +--:(srv6-stack)
|     |        |  +--rw srv6-stack*           fpcbase:segment-id
|     |        +--:(tunnel-info)
|     |           +--rw tunnel-info
|     |              +--rw tunnel-local-address?    inet:ip-address
|     |              +--rw tunnel-remote-address?   inet:ip-address
|     |              +--rw mtu-size?                uint32
|     |              +--rw tunnel?                  identityref
|     |              +--rw payload-type?            enumeration
|     |              +--rw gre-key?                 uint32
|     |              +--rw gtp-tunnel-info
|     |              |  +--rw local-tunnel-identifier?    uint32
|     |              |  +--rw remote-tunnel-identifier?   uint32
|     |              |  +--rw sequence-numbers-enabled?   boolean
|     |              +--rw ebi?                     fpcbase:ebi-type
|     |              +--rw lbi?                     fpcbase:ebi-type
|     +--:(nexthop)
|     |  +--rw nexthop
|     |     +--rw (next-hop-value)?
|     |        +--:(ip-address)
|     |        |  +--rw ip-address?           inet:ip-address
|     |        +--:(mac-address)
|     |        |  +--rw mac-address?          ytypes:mac-address
|     |        +--:(service-path)
|     |        |  +--rw service-path?   fpcbase:fpc-service-path-id
|     |        +--:(mpls-path)
```

```
|     |        |  +--rw mpls-path?           fpcbase:fpc-mpls-label
|     |        +--:(nsh)
|     |        |  +--rw nsh?                string
|     |        +--:(interface)
|     |        |  +--rw interface?          uint16
|     |        +--:(segment-identifier)
|     |        |  +--rw segment-identifier? fpcbase:segment-id
|     |        +--:(mpls-label-stack)
|     |        |  +--rw mpls-label-stack*   fpcbase:fpc-mpls-label
|     |        +--:(mpls-sr-stack)
|     |        |  +--rw mpls-sr-stack*      fpcbase:fpc-mpls-label
|     |        +--:(srv6-stack)
|     |        |  +--rw srv6-stack*         fpcbase:segment-id
|     |        +--:(tunnel-info)
|     |           +--rw tunnel-info
|     |              +--rw tunnel-local-address?    inet:ip-address
|     |              +--rw tunnel-remote-address?   inet:ip-address
|     |              +--rw mtu-size?                uint32
|     |              +--rw tunnel?                  identityref
|     |              +--rw payload-type?            enumeration
|     |              +--rw gre-key?                 uint32
|     |              +--rw gtp-tunnel-info
|     |              |  +--rw local-tunnel-identifier?    uint32
|     |              |  +--rw remote-tunnel-identifier?   uint32
|     |              |  +--rw sequence-numbers-enabled?   boolean
|     |              +--rw ebi?                     fpcbase:ebi-type
|     |              +--rw lbi?                     fpcbase:ebi-type
|     +--:(qos)
|        +--rw trafficclass?                  inet:dscp
|        +--rw per-mn-agg-max-dl?
                 qos-pmip:Per-MN-Agg-Max-DL-Bit-Rate-Value
|        +--rw per-mn-agg-max-ul?
                 qos-pmip:Per-MN-Agg-Max-UL-Bit-Rate-Value
|        +--rw per-session-agg-max-dl
|        |  +--rw max-rate        uint32
|        |  +--rw service-flag    boolean
|        |  +--rw exclude-flag    boolean
|        +--rw per-session-agg-max-ul
|        |  +--rw max-rate        uint32
|        |  +--rw service-flag    boolean
|        |  +--rw exclude-flag    boolean
|        +--rw priority-level                 uint8
|        +--rw preemption-capability           enumeration
|        +--rw preemption-vulnerability        enumeration
|        +--rw agg-max-dl?
                 qos-pmip:Aggregate-Max-DL-Bit-Rate-Value
|        +--rw agg-max-ul?
                 qos-pmip:Aggregate-Max-UL-Bit-Rate-Value
```

```
|        +--rw gbr-dl?
                 qos-pmip:Guaranteed-DL-Bit-Rate-Value
|        +--rw gbr-ul?
                 qos-pmip:Guaranteed-UL-Bit-Rate-Value
|        +--rw qci?
                 fpcbase:fpc-qos-class-identifier
|        +--rw ue-agg-max-bitrate?            uint32
|        +--rw apn-ambr?                      uint32

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

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

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

```
|  |        +--rw peer-service-group-key*   fpc:fpc-identity
|  +--rw dpn* [dpn-key]
|  |  +--rw dpn-key                          fpc:fpc-identity
|  |  +--rw dpn-name?                        string
|  |  +--rw dpn-resource-mapping-reference?  string
|  |  +--rw domain-key                       fpc:fpc-identity
|  |  +--rw service-group-key*               fpc:fpc-identity
|  |  +--rw interface* [interface-key]
|  |  |  +--rw interface-key        fpc:fpc-identity
|  |  |  +--rw interface-name?      string
|  |  |  +--rw role?                identityref
|  |  |  +--rw protocol*            identityref
|  |  |  +--rw interface-configuration* [index]
|  |  |     +--rw (policy-configuration-value)?
|  |  |           |  ...
|  |  +--rw dpn-policy-configuration* [policy-template-key]
|  |     +--rw policy-template-key     fpc:fpc-identity
|  |     +--rw policy-configuration* [index]
|  |        +--rw index    uint16
|  |          +--rw (policy-configuration-value)?
|  |               |  ...
|  +--rw domain* [domain-key]
|  |  +--rw domain-key        fpc:fpc-identity
|  |  +--rw domain-name?      string
|  |  +--rw domain-policy-configuration* [policy-template-key]
|  |     +--rw policy-template-key     fpc:fpc-identity
|  |     +--rw policy-configuration* [index]
|  |             |         ...
|  +--rw dpn-checkpoint
|  |  +--rw basename?          fpc:fpc-identity
|  |  +--rw base-checkpoint?   string
|  +--rw service-group-checkpoint
|  |  +--rw basename?          fpc:fpc-identity
|  |  +--rw base-checkpoint?   string
|  +--rw dpn-checkpoint
|  |  +--rw basename?                   fpc:fpc-identity
|  |  +--rw base-checkpoint?            string
+--rw policy-information-model
|  +--rw action-template* [action-template-key]
|  |  +--rw action-template-key        fpc:fpc-identity
|  |  +--rw (action-value)
|  |  |                   ...
|  |  +--rw extensible?                boolean
|  |  +--rw static-attributes*         string
|  |  +--rw mandatory-attributes*      string
|  |  +--rw entity-state?              enumeration
|  |  +--rw version?                   uint32
|  +--rw descriptor-template* [descriptor-template-key]
```

```
|  |  +--rw descriptor-template-key         fpc:fpc-identity
|  |  +--rw (descriptor-value)
|  |  |                    ...
|  |  +--rw extensible?                     boolean
|  |  +--rw static-attributes*              string
|  |  +--rw mandatory-attributes*           string
|  |  +--rw entity-state?                   enumeration
|  |  +--rw version?                        uint32
|  +--rw rule-template* [rule-template-key]
|  |  +--rw rule-template-key           fpc:fpc-identity
|  |  +--rw descriptor-match-type       enumeration
|  |  +--rw descriptor-configuration* [descriptor-template-key]
|  |  |  +--rw descriptor-template-key    fpc:fpc-identity
|  |  |  +--rw direction?                 rfc5777:direction-type
|  |  |  +--rw setting?                        <anydata>
|  |  |  +--rw attribute-expression* [index]
|  |  |     +--rw index                           uint16
|  |  |     +--rw (descriptor-value)
|  |  |     |  ...
|  |  +--rw action-configuration* [action-order]
|  |  |  +--rw action-order            uint32
|  |  |  +--rw action-template-key     fpc:fpc-identity
|  |  |  +--rw setting?                        <anydata>
|  |  |  +--rw attribute-expression* [index]
|  |  |     +--rw index                     uint16
|  |  |     +--rw (action-value)
|  |  |     |  ...
|  |  +--rw extensible?                 boolean
|  |  +--rw static-attributes*          string
|  |  +--rw mandatory-attributes*       string
|  |  +--rw entity-state?               enumeration
|  |  +--rw version?                    uint32
|  |  +--rw rule-configuration* [index]
|  |     +--rw index    uint16
|  |         +--rw (policy-configuration-value)?
|  |           |  ...
|  +--rw policy-template* [policy-template-key]
|  |  +--rw policy-template-key     fpc:fpc-identity
|  |  +--rw rule-template* [precedence]
|  |  |  +--rw precedence           uint32
|  |  |  +--rw rule-template-key    fpc:fpc-identity
|  |  +--rw extensible?             boolean
|  |  +--rw static-attributes*      string
|  |  +--rw mandatory-attributes*   string
|  |  +--rw entity-state?           enumeration
|  |  +--rw version?                uint32
|  |  +--rw policy-configuration* [index]
|  |      ...
```

```
|  +--rw basename?              fpc:fpc-identity
|  +--rw base-checkpoint?       string
+--rw mobility-context* [mobility-context-key]
|  +--rw mobility-context-key    fpc:fpc-identity
|  +--rw delegating-ip-prefix*   inet:ip-prefix
|  +--rw parent-context?         fpc:fpc-identity
|  +--rw child-context*          fpc:fpc-identity
|  +--rw mobile-node
|  |  +--rw ip-address*    inet:ip-address
|  |  +--rw imsi?          fpcbase:imsi-type
|  |  +--rw mn-policy-configuration* [policy-template-key]
|  |     +--rw policy-template-key     fpc:fpc-identity
|  |     +--rw policy-configuration* [index]
|  |        ...
|  +--rw domain
|  |  +--rw domain-key?        fpc:fpc-identity
|  |  +--rw domain-policy-configuration* [policy-template-key]
|  |     +--rw policy-template-key     fpc:fpc-identity
|  |     +--rw policy-configuration* [index]
|  |        ...
|   +--rw dpn* [dpn-key]
|      +--rw dpn-key              fpc:fpc-identity
|      +--rw dpn-policy-configuration* [policy-template-key]
|      |  +--rw policy-template-key     fpc:fpc-identity
|      |  +--rw policy-configuration* [index]
|      |     ...
|      +--rw role?                identityref
|      +--rw service-data-flow* [identifier]
|         +--rw identifier           uint32
|         +--rw service-group-key?   fpc:fpc-identity
|         +--rw interface* [interface-key]
|         |  +--rw interface-key    fpc:fpc-identity
|         +--rw service-data-flow-policy-
                configuration* [policy-template-key]
|            +--rw policy-template-key     fpc:fpc-identity
|            +--rw policy-configuration* [index]
|               ...
+--rw monitor* [monitor-key]
  +--rw extensible?             boolean
  +--rw static-attributes*      string
  +--rw mandatory-attributes*   string
  +--rw entity-state?           enumeration
  +--rw version?                uint32
  +--rw monitor-key             fpc:fpc-identity
  +--rw target?                 string
  +--rw deferrable?             boolean
  +--rw (configuration)
     +--:(period)
```

```
          |  +--rw period?           uint32
          +--:(threshold-config)
          |  +--rw low?              uint32
          |  +--rw hi?               uint32
          +--:(schedule)
          |  +--rw schedule?         uint32
          +--:(event-identities)
          |  +--rw event-identities*       identityref
          +--:(event-ids)
             +--rw event-ids*        uint32

     rpcs:
       +---x configure
       |  +---w input
       |  |  +---w client-id          fpc:client-identifier
       |  |  +---w execution-delay?   uint32
       |  |  +---w yang-patch
       |  |     +---w patch-id    string
       |  |     +---w comment?    string
       |  |     +---w edit* [edit-id]
       |  |        +---w edit-id         string
       |  |        +---w operation       enumeration
       |  |        +---w target          target-resource-offset
       |  |        +---w point?          target-resource-offset
       |  |        +---w where?          enumeration
       |  |        +---w value?          <anydata>
       |  |        +---w reference-scope?   fpc:ref-scope
       |  |        +---w command-set
       |  |           +---w (instr-type)?
       |  |              +--:(instr-3gpp-mob)
       |  |              |  +---w instr-3gpp-mob? fpcbase:threegpp-instr
       |  |              +--:(instr-pmip)
       |  |                 +---w instr-pmip?       pmip-commandset
       |  +--ro output
       |     +--ro yang-patch-status
       |        +--ro patch-id       string
       |        +--ro (global-status)?
       |        |  +--:(global-errors)
       |        |  |  +--ro errors
       |        |  |     +--ro error*
       |        |  |        +--ro error-type       enumeration
       |        |  |        +--ro error-tag        string
       |        |  |        +--ro error-app-tag?   string
       |        |  |        +--ro error-path?      instance-identifier
       |        |  |        +--ro error-message?   string
       |        |  |        +--ro error-info?      <anydata>
       |        |  +--:(ok)
       |        |     +--ro ok?            empty
```

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

```
|  |        +--:(event-identities)
|  |        |  +---w event-identities*       identityref
|  |        +--:(event-ids)
|  |           +---w event-ids*              uint32
|  +--ro output
|     +--ro operation-id     uint64
|     +--ro (edit-status-choice)?
|        +--:(ok)
|        |  +--ro ok?       empty
|        +--:(errors)
|           +--ro errors
|              +--ro error*
|                 +--ro error-type       enumeration
|                 +--ro error-tag        string
|                 +--ro error-app-tag?   string
|                 +--ro error-path?      instance-identifier
|                 +--ro error-message?   string
|                 +--ro error-info?      <anydata>
+---x deregister_monitor
|  +---w input
|  |  +---w client-id          fpc:client-identifier
|  |  +---w execution-delay?   uint32
|  |  +---w operation-id              uint64
|  |  +---w monitor* [monitor-key]
|  |     +---w monitor-key    fpc:fpc-identity
|  |     +---w send_data?     boolean
|  +--ro output
|     +--ro operation-id     uint64
|     +--ro (edit-status-choice)?
|        +--:(ok)
|        |  +--ro ok?       empty
|        +--:(errors)
|           +--ro errors
|              +--ro error*
|                 +--ro error-type       enumeration
|                 +--ro error-tag        string
|                 +--ro error-app-tag?   string
|                 +--ro error-path?      instance-identifier
|                 +--ro error-message?   string
|                 +--ro error-info?      <anydata>
+---x probe
   +---w input
   |  +---w client-id          fpc:client-identifier
   |  +---w execution-delay?   uint32
   |  +---w operation-id              uint64
   |  +---w monitor* [monitor-key]
   |     +---w monitor-key    fpc:fpc-identity
   +--ro output
```

```
            +--ro operation-id     uint64
            +--ro (edit-status-choice)?
               +--:(ok)
               |  +--ro ok?       empty
               +--:(errors)
                  +--ro errors
                     +--ro error*
                        +--ro error-type       enumeration
                        +--ro error-tag        string
                        +--ro error-app-tag?   string
                        +--ro error-path?      instance-identifier
                        +--ro error-message?   string
                        +--ro error-info?      <anydata>

     notifications:
       +---n config-result-notification
       |  +--ro yang-patch-status
       |  |  +--ro patch-id       string
       |  |  +--ro (global-status)?
       |  |  |  +--:(global-errors)
       |  |  |  |  +--ro errors
       |  |  |  |     +--ro error*
       |  |  |  |        +--ro error-type       enumeration
       |  |  |  |        +--ro error-tag        string
       |  |  |  |        +--ro error-app-tag?   string
       |  |  |  |        +--ro error-path?      instance-identifier
       |  |  |  |        +--ro error-message?   string
       |  |  |  |        +--ro error-info?      <anydata>
       |  |  |  +--:(ok)
       |  |  |     +--ro ok?            empty
       |  |  +--ro edit-status
       |  |     +--ro edit* [edit-id]
       |  |        +--ro edit-id    string
       |  |        +--ro (edit-status-choice)?
       |  |           +--:(ok)
       |  |           |  +--ro ok?        empty
       |  |           +--:(errors)
       |  |              +--ro errors
       |  |                 +--ro error*
       |  |                    +--ro error-type       enumeration
       |  |                    +--ro error-tag        string
       |  |                    +--ro error-app-tag?   string
       |  |                    +--ro error-path?
       |  |                            instance-identifier
       |  |                    +--ro error-message?   string
       |  |                    +--ro error-info?      <anydata>
       |  +--ro subsequent-edit* [edit-id]
       |     +--ro edit-id      string
```

```
|     +--ro operation    enumeration
|     +--ro target       ypatch:target-resource-offset
|     +--ro point?       ypatch:target-resource-offset
|     +--ro where?       enumeration
|     +--ro value?       <anydata>
+---n notify
   +--ro notification-id?   uint32
   +--ro timestamp?         uint32
   +--ro report* [monitor-key]
      +--ro monitor-key                fpc:fpc-identity
      +--ro trigger?                   identityref
      +--ro (value)?
         +--:(dpn-candidate-available)
         |  +--ro node-id?                    inet:uri
         |  +--ro supported-interface-list* [role-key]
         |     +--ro role-key    identityref
         +--:(dpn-unavailable)
         |  +--ro dpn-id?                     fpc:fpc-identity
         +--:(report-value)
            +--ro report-value?             <anydata>
```

Figure 38: YANG FPC Agent Tree

Appendix C. Change Log

C.1. Changes since Version 09

The following changes have been made since version 09

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

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

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

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

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

The Unassigned DPN is used to track requests vs. those that are installed, i.e. Agent assignment of Policy is supported.

The Topology system supports selection information by ServiceGroup or ServiceEndpoint.

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

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

RFC 5777 Classifiers have been added.

All operations have a common error format.

C.2. Changes since Version 10

The following changes have been made since version 10

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

Service-Group resides under the Topology-Information-Mode

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

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

Monitor lifecycle, policy and policy installation examples added.

Authors' Addresses

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

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

Lyle Bertz
6220 Sprint Parkway
Overland Park KS, 66251
USA

Email: lylebe551144@gmail.com

Marco Liebsch
NEC Laboratories Europe
NEC Europe Ltd.
Kurfuersten-Anlage 36
D-69115 Heidelberg
Germany

Phone: +49 6221 4342146
Email: liebsch@neclab.eu

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

Email: sgundave@cisco.com

Danny Moses

Email: danny.moses@intel.com

Charles E. Perkins
Futurewei Inc.
2330 Central Expressway
Santa Clara, CA 95050
USA

Phone: +1-408-330-4586
Email: charliep@computer.org

DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2018

S. Matsushima
SoftBank
C. Filsfils
M. Kohno
P. Camarillo
Cisco Systems, Inc.
D. Voyer
Bell Canada
C. Perkins
Futurewei

March 5, 2018

Segment Routing IPv6 for Mobile User Plane
draft-ietf-dmm-srv6-mobile-uplane-01

Abstract

This document discusses the applicability of SRv6 (Segment Routing IPv6) to user-plane of mobile networks. The source routing capability and the network programming nature of SRv6, accomplish mobile user-plane functions in a simple manner. The statelessness and the ability to control underlying layer will be even more beneficial to the mobile user-plane, in terms of providing flexibility and SLA control for various applications. It also simplifies the network architecture by eliminating the necessity of tunnels, such as GTP-U [TS.29281], PMIP [RFC5213], Mac-in-Mac, MPLS, and so on. In addition, Segment Routing provides an enhanced method for network slicing, which is briefly introduced by this document.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 6, 2018.

Table of Contents

1. Introduction

In mobile networks, mobility management systems provide connectivity while mobile nodes move around. While the control-plane of the system signals movements of a mobile node, user-plane establishes tunnel between the mobile node and anchor node over IP based backhaul and core networks.

This document discusses the applicability of SRv6 (Segment Routing IPv6) to those mobile networks. SRv6 provides source routing to networks where operators can explicitly indicate a route for the packets from and to the mobile node. SRv6 endpoint nodes perform the roles of anchor of mobile user-plane.

2. Conventions and Terminology

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

SRH is the abbreviation for the Segment Routing Header. We assume that the SRH may be present multiple times inside each packet.

NH is the abbreviation of the IPv6 next-header field.

NH=SRH means that the next-header field is 43 with routing type 4.

When there are multiple SRHs, they must follow each other: the next-header field of all SRH, except the last one, must be SRH.

The effective next-header (ENH) is the next-header field of the IP header when no SRH is present, or is the next-header field of the last SRH.

In this version of the document, we assume that there is no other extension header than the SRH. This will be lifted in future versions of the document.

SID: A Segment Identifier which represents a specific segment in segment routing domain. The SID type used in this document is IPv6 address (also referenced as SRv6 Segment or SRv6 SID).

A SID list is represented as <S1, S2, S3> where S1 is the first SID to visit, S2 is the second SID to visit and S3 is the last SID to visit along the SR path.

(SA,DA) (S3, S2, S1; SL) represents an IPv6 packet with:

- IPv6 header with source and destination addresses respectively SA and DA and next-header is SRH
- SRH with SID list <S1, S2, S3> with SegmentsLeft = SL
- Note the difference between the <> and () symbols: <S1, S2, S3> represents a SID list where S1 is the first SID and S3 is the last SID. (S3, S2, S1; SL) represents the same SID list but encoded in the SRH format where the rightmost SID in the SRH is the first SID and the leftmost SID in the SRH is the last SID. When referring to an SR policy in a high-level use-case, it is simpler to use the <S1, S2, S3> notation. When referring to an illustration of the detailed behavior, the (S3, S2, S1; SL) notation is more convenient.
- The payload of the packet is omitted.

SRH[SL] represents the SID pointed by the SL field in the first SRH. In our example, SRH[2] represents S1, SRH[1] represents S2 and SRH[0] represents S3.

FIB is the abbreviation for the forwarding table. A FIB lookup is a lookup in the forwarding table. When a packet is intercepted on a wire, it is possible that SRH[SL] is different from the DA.

3. Motivation

Every day mobility networks are getting more challenging to operate: on one hand, traffic is constantly growing, and latency requirements are more strict; on the other-hand, there are new use-cases like NFV that are also challenging network management.

Problem comes from the fact that the current architecture of mobile networks is agnostic to the underlying transport. Indeed, it rigidly fragments the user-plane into radio access, core and service networks and connects them by tunneling techniques through the user-plane roles such as access and anchor nodes. Such agnosticism and rigidness make it difficult for the operator to optimize and operate the data-path.

While the mobile network industry has been trying to solve those problems, applications have shifted to use IPv6, and network operators have started adopting IPv6 as their IP transport as well. SRv6, the IPv6 instantiation of Segment Routing [I-D.ietf-spring-segment-routing], integrates both the application

data-path and the underlying transport layer into one single protocol, allowing operators to optimize the network in a simplified manner and removing state from the network.

Further on, SRv6 introduces the notion of network-programming [I-D.filsfils-spring-srv6-network-programming], that applied to mobility fulfils the user-plane functions of mobility management. SRv6 takes advantage of underlying transport awareness and flexibility to deploy mobility user-plane functions in an optimized manner. Those are the motivations to adopt SRv6 for mobile user-plane.

4. Reference Architecture

This section describes a reference architecture and possible deployment scenarios.

Figure 1 shows a reference architecture, based on 5G packet core architecture [TS.23501].

Please note that all the user-plane described in this document does not depend on any specific architecture. This architecture is just used as a reference based on the latest 3GPP standards at the time of writing this draft. Other type of architectures can be seen in [I-D.gundavelli-dmm-mfa] and [WHITEPAPER-5G-UP].

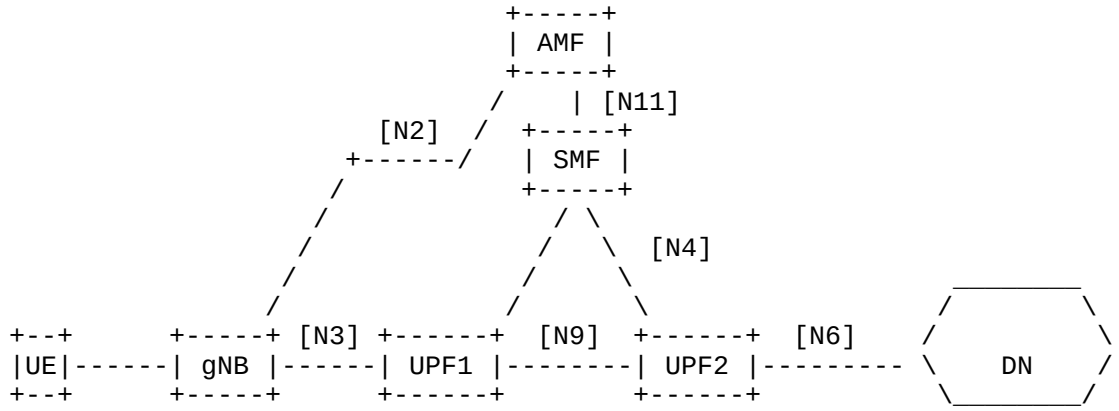


Figure 1: Reference Architecture

- UE : User Equipment
- gNB : gNodeB
- UPF : User Plane Function
  - UPF1: Interfaces N3 and N9

    * UPF2: Interfaces N9 and N6
    * Note: For simplicity we don’t depict a UPF that is only connected to N9 interfaces, although the techniques described in this document are also valid in such case.
- SMF : Session Management Function
- AMF : Access and Mobility Management Function
- DN : Data Network e.g. operator services, Internet access

A session from an UE gets assigned to an UPF. Sometimes more than one UPF may be used for providing a certain kind of richer service functions. UE gets its IP address from the DHCP block of its UPF. The UPF advertises the IP address block towards the Internet ensuring that return traffic is routed to the right UPF.

5. User-plane behaviors

This section describes the mobile user-plane behaviors using SRv6.

In order to simplify the SRv6 adoption, we present two different "modes" that vary with respect the SRv6 SID allocation. The first one is the "Traditional mode", which inherits the traditional mobile user-plane. In this mode there is no change to mobility networks architecture, except for the pure replacement of GTP-U [TS.29281] for SRv6.

The second mode is the "Enhanced mode", which aggregates the mobile sessions and allocates SID on a per policy basis. The benefit of the latter is that the SR policy contains SIDs for Traffic Engineering and VNFs. Both of these modes assume both the gNB and UPFs are SR-aware (N3 and N9 interfaces are SRv6).

Additionally, we introduce a new "Enhanced mode with unchanged gNB GTP behavior". This mode consists of two mechanisms for interworking with legacy access networks -interface N3 unmodified-. One of these mechanism is designed to interwork with legacy gNBs using GTP/IPv4. The second method is designed to interwork with legacy gNBs using GTP/IPv6.

This section makes reference to already existing SRv6 functions defined in [I-D.filsfils-spring-srv6-network-programming] as well as new SRv6 functions designed for the mobile userplane. The new SRv6 functions are detailed in the Section 6.

5.1. Traditional mode (formerly Basic mode)

In the traditional mode, we assume that mobile user-plane functions are the same as existing ones except the use of SRv6 as the data

plane instead of GTP-U. No impact to the rest of mobile system should be expected.

In the traditional mobile network, an UE session is mapped 1-for-1 with a specific GTP tunnel (TEID). This 1-for-1 mapping is replicated here to replace the GTP encaps with the SRv6 encaps, while not changing anything else.

This mode minimizes the changes required to the entire system and it is a good starting point for forming the common basis. Note that in this mode the TEID is embedded in each SID.

Our reference topology is shown in Figure 2. In this mode we assume that the gNB and the UPFs are SR-aware.

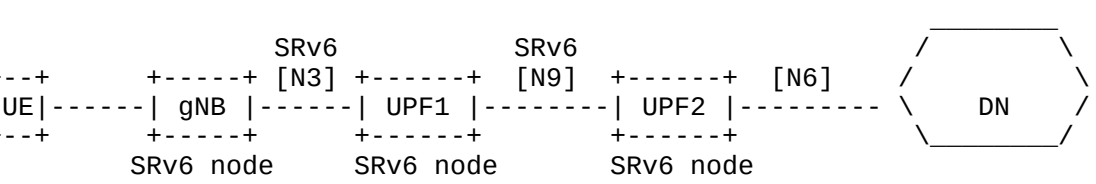


Figure 2: Traditional mode - Reference topology

5.1.1. Packet flow - Uplink

The uplink packet flow is the following:

```
UE_out  : (A,Z)
gNB_out : (gNB, U1::1) (A,Z)     -> T.Encaps.Reduced <U1::1>
UPF1_out: (gNB, U2::1) (A,Z)     -> End.MAP
UPF2_out: (A,Z)                  -> End.DT4 or End.DT6
```

The UE packet arrives to the gNB. The gNB performs a T.Encaps.Reduced operations. Since there is only one SID, there is no need to push an SRH. gNB only adds an outer IPv6 header with IPv6 DA U1::1. U1::1 represents an anchoring SID specific for that session at UPF1. The SID U1::1 is retrieved through the existing control plane (N2 interface).

Upon packet arrival on UPF1, the SID U1::1 is a local End.MAP function. This function maps the SID with the next anchoring point and replaces U1::1 by U2::1, that belongs to the next anchoring point.

Upon packet arrival on UPF2, the SID U2::1 corresponds to an End.DT function. UPF2 decapsulates the packet, performs a lookup in a specific table and forwards the packet towards the data network.

5.1.2. Packet flow - Downlink

The downlink packet flow is the following:

```
UPF2_in : (Z,A)
UPF2_out: (U2::, U1::1) (Z,A)    -> T.Encaps.Reduced <U1::1>
UPF1_out: (U2::, gNB::1) (Z,A)      -> End.MAP
gNB_out : (Z,A)                  -> End.DX4 or End.DX6
```

When the packet arrives to the UPF2, the UPF2 will map that particular flow into a UE session. This UE session is associated with the policy <U1::1>. The UPF2 performs a T.Encaps.Reduced operation, encapsulating the packet into a new IPv6 header with no SRH since there is only one SID.

Upon packet arrival on UPF1, the SID U1::1 is a local End.MAP function. This function maps the SID with the next anchoring point and replaces U1::1 by gNB::1, that belongs to the next anchoring point.

Upon packet arrival on gNB, the SID gNB::1 corresponds to an End.DX4/ End.DX6 function. The gNB will decapsulates the packet, removing the IPv6 header and all it’s extensions headers and will forward the traffic towards the UE.

5.1.3. IPv6 user-traffic

For IPv6 user-traffic it is RECOMMENDED to perform encapsulation. However based on local policy, a service provider MAY choose to do SRH insertion. The main benefit is a lower overhead. In such case, the functions used are T.Insert.Red at gNB, End.MAP at UPF1 and End.T at UPF2 on Uplink, T.Insert.Red at UPF2, End.MAP at UPF1 and End.X at gNB on Downlink.

5.2. Enhanced Mode (formerly Aggregate mode)

This mode improves the scalability. In addition, it provides key improvements in terms of traffic steering and service chaining, thanks to the use of an SR policy of multiple SIDs, instead of single one in the Traditional mode.

Key points:

- Several UE share the same SR Policy (and it’s composing SID)
- The SR policy MAY include SIDs for traffic engineering and service chaining on top of the UPF anchor.

The gNB control-plane (N2 interface) is unchanged, specifically a single IPv6 address is given to the gNB.

- The gNB MAY resolve the IP address into a SID list through a mechanism like PCEP, DNS-lookup, small augment for LISP control-plane, etc.

Our reference topology is shown in Figure 3. In this mode we assume that the gNB and the UPF are SR-aware. We also assume that we have two services segments, S1 and C1. S1 represents a VNF in the network, and C1 represents a constraint path on a router over which we are going to perform Traffic Engineering. Note that S1 and C1 belong to the underlay and don't have an N4 interface. For this reason we don't consider them UPFs.

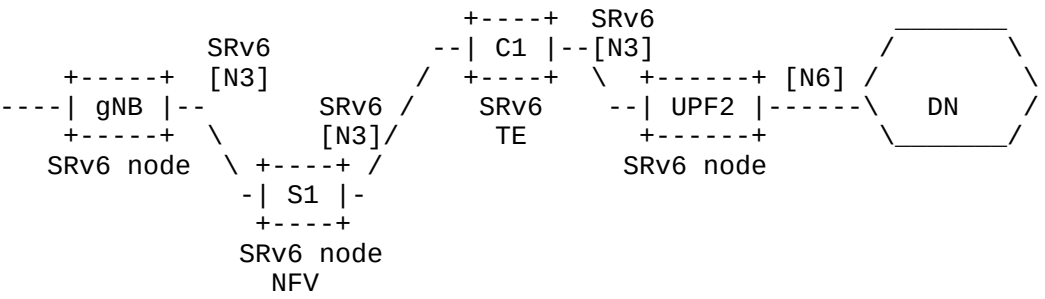


Figure 3: Enhanced mode - Reference topology

5.2.1. Packet flow - Uplink

The uplink packet flow is the following:

```
UE_out  : (A,Z)
gNB_out : (gNB, S1)(U2::1, C1; SL=2)(A,Z)-> T.Encaps.Red<S1,C1,U2::1>
S1_out  : (gNB, C1)(U2::1, C1; SL=1 (A,Z)
C1_out  : (gNB, U2::1)(A,Z)              -> PSP
UPF2_out: (A,Z)                          -> End.DT4 or End.DT6
```

UE sends its packet (A,Z) on a specific bearer session to its gNB. gNB's CP associates that session from the UE(A) with the IPv6 address B and GTP TEID T. gNB's CP does a lookup on B (by reverseDNS, LISP, etc.) to find the related SID list <S1, C1, U2::1>.

Once the packet leaves the gNB, it already contains all the segments of the SR policy. This SR policy contains segments for traffic engineering (C1) and for service chaining (S1).

The nodes S1 and C1 perform their related Endpoint functionality and forward.

When the packet arrives to UPF2, the active segment (U2::1) is an End.DT4/6 which performs the decapsulation (removing the IPv6 header with all it's extension headers) and forward towards the data network.

Note that in case several APNs are using duplicated IPv4 private address spaces, then the aggregated SR policies are unique per APNs.

5.2.2. Packet flow - Downlink

The downlink packet flow is the following:

```
UPF2_in : (Z,A)                                -> UPF2 maps the flow w/
                                                  SID list <C1,S1, gNB>
UPF2_out: (U2::1, C1)(gNB, S1; SL=2)(Z,A)      -> T.Encaps.Red
C1_out  : (U2::1, S1)(gNB, S1; SL=1)(Z,A)
S1_out  : (U2::1, gNB)(Z,A)                    -> PSP
gNB_out : (Z,A)                                -> End.DX4 or End.DX6
```

When the packet arrives to the UPF2, the UPF2 will map that particular flow into a UE session. This UE session is associated with the policy <C1, S1, gNB>. The UPF2 performs a T.Encaps.Reduced operation, encapsulating the packet into a new IPv6 header with its corresponding SRH.

The nodes C1 and S1 perform their related Endpoint processing.

Once the packet arrives to the gNB, the IPv6 DA corresponds to an End.DX4 or End.DX6 (depending on the underlying traffic). The gNB will decapsulate the packet, removing the IPv6 header and all it's extensions headers and will forward the traffic towards the UE.

5.2.3. IPv6 user-traffic

For IPv6 user-traffic it is RECOMMENDED to perform encapsulation. However based on local policy, a service provider MAY choose to do SRH insertion. The main benefit is a lower overhead. In such case, the functions used are T.Insert.Red at gNB and End.T at UPF2 on Uplink, T.Insert.Red at UPF2 and End.X at gNB on Downlink.

5.3. Enhanced mode with unchanged gNB GTP behavior

In this section we introduce two mechanisms for interworking with legacy gNBs that still use GTP. One of the mechanisms is valid for IPv4 while the other for IPv6.

In this scenario, it is assumed that gNB does not support SRv6. It just supports GTP encapsulation over IPv4 or IPv6. Hence in order to achieve interworking we are going to add a new SR Gateway (SRGW-UPF1) entity. This SRGW is going to map the GTP traffic into SRv6. Note that the SR GW is not an anchor point.

The SRGW maintains very little state on it. For this reason, both of these methods (IPv4 and IPv6) scale to millions of UEs.

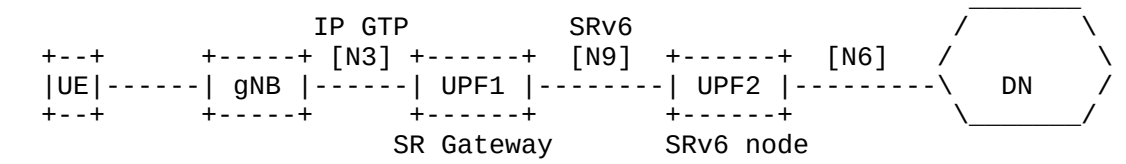


Figure 4: Reference topology for interworking

5.3.1. Interworking with IPv6 GTP

In this interworking mode we assume that the gNB is using GTP over IPv6 in the N3 interface

Key points:

- gNB is unchanged (control-plane or user-plane) and encaps into GTP (N3 interface is not modified).
- 5G Control-Plane (N2 interface) is unmodified: 1 IPv6 address (i.e. a BSID at the SRGW)
- SRGW removes GTP, finds SID list related to DA, add SRH with the SID list.
- There is NO state for the downlink at the SRGW.
- There is simple state in the uplink at the SRGW (leveraging the enhanced mode results in few SR policies on this node. A SR policy can be shared across UEs).
- As soon as the packet leaves the gNB (uplink), the traffic is SR-routed. This simplifies considerably network slicing [I-D.hegdeppsenak-isis-sr-flex-algo].
- In the uplink, we use the IPv6 DA BSID to steer the traffic into an SR policy when it arrives at the SRGW-UPF1-.

Our reference topology is shown in Figure 5. In this mode we assume that the gNB is an unmodified gNB using IPv6/GTP. The UPFs are SR-aware. Also, as explained before, we introduce a new SRGW entity that is going to map the IPv6/GTP traffic to SRv6.

We also assume that we have two service segment, S1 and C1. S1 represents a VNF in the network, and C1 represents a router over which we are going to perform Traffic Engineering.

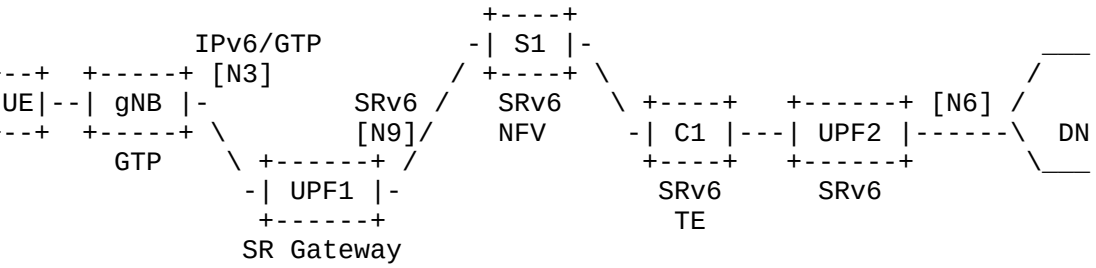


Figure 5: Enhanced mode with unchanged gNB IPv6/GTP behavior

5.3.1.1. Packet flow - Uplink

The uplink packet flow is the following:

```
UE_out  : (A,Z)
gNB_out : (gNB, B)(GTP: TEID T)(A,Z)       -> Interface N3 unmodified
                                              (IPv6/GTP)
SRGW_out: (SRGW, S1)(U2::1, C1; SL=2)(A,Z) -> B is an End.M.GTP6.D
                                              SID at the SRGW
S1_out  : (SRGW, C1)(U2::1, C1; SL=1)(A,Z)
C1_out  : (SRGW, U2::1)(A,Z)               -> PSP
UPF2_out: (A,Z)                            -> End.DT4 or End.DT6
```

The UE sends a packet destined to Z towards the gNB on a specific bearer for that session. The gNB, which is unmodified, encapsulates the packet into a new IPv6, UDP and GTP headers. The IPv6 DA B, and the GTP TEID T are the ones received in the N2 interface.

The IPv6 address that was signalled over the N2 interface for that UE session, B, is now the IPv6 DA. B is an SRv6 Binding SID instantiated at the SRGW. Hence the packet, will be routed up to the SRGW.

When the packet arrives at the SRGW, the SRGW realises that B is an End.M.GTP6.D BindingSID. Hence, the SRGW will remove the IPv6, UDP and GTP headers, and will push a new IPv6 header with its own SRH containing the SIDs bound to the SR policy associated with this BindingSID.

The nodes S1 and C1 perform their related Endpoint functionality and forward.

When the packet arrives to UPF2, the active segment is (U2::1) which bound to End.DT4/6 which is going to perform the decapsulation (removing the outer IPv6 header with all it's extension headers) and forward towards the data network.

5.3.1.2. Packet flow - Downlink

The downlink packet flow is the following:

```
UPF2_in : (Z,A)                              -> UPF2 maps the flow with
                                                <C1, S1, SRGW::TEID,gNB>
UPF2_out: (U2::1, C1)(gNB, SRGW::TEID, S1; SL=3)(Z,A) -> T.Encaps.Red
C1_out  : (U2::1, S1)(gNB, S1; SL=2)(Z,A)
S1_out  : (U2::1, SRGW::TEID)(gNB, SRGW::TEID, S1, SL=1)(Z,A)
SRGW_out: (SRGW, gNB)(GTP: TEID=T)(Z,A)   -> SRGW/96 is End.M.GTP6.E
gNB_out : (Z,A)
```

When a packet destined to A arrives at the UPF2, the UPF2 performs a lookup in the associated table to A and finds the SID list <C1, S1, SRGW::TEID, gNB>. The UPF2 performs a T.Encaps.Reduced operation, encapsulating the packet into a new IPv6 header with its corresponding SRH.

The nodes C1 and S1 perform their related Endpoint processing.

Once the packet arrives to the SRGW, the SRGW realizes the active SID is an End.M.GTP6.E function. The SRGW removes the IPv6 header and all it's extensions headers. The SRGW generates an IPv6, UDP and GTP headers. The new IPv6 DA is the gNB which is the last SID in the received SRH. The TEID in the generated GTP header is the arguments of the received End.M.GTP6.E SID. The SRGW pushes the headers to the packet and forwards the packet towards the gNB.

Once the packet arrives to the gNB, the packet is a regular IPv6/GTP packet. The gNB looks for the specific radio bearer for that TEID and forward it on the bearer. This gNB behavior is not modified from current and previous generations.

5.3.1.3. Scalability

For the downlink traffic, the SRGW is stateless. All the state is in the SRH imposed by the UPF2. The UPF2 must have the UE states as the session anchor point.

For the uplink traffic, the state at the SRGW does not necessarily need to be per UE session basis. A state of SR policy of which state can be shared among UE's. Hence it is possible to deploy SRGW in

very scalable way compared to hold millions of states per UE session basis.

5.3.1.4. IPv6 user-traffic

For IPv6 user-traffic it is RECOMMENDED to perform encapsulation. However based on local policy, a service provider MAY choose to do SRH insertion. The main benefit is a lower overhead.

5.3.2. Interworking with IPv4 GTP

In this interworking mode we assume that the gNB is using GTP over IPv4 in the N3 interface

Key points:

- gNB is unchanged and encaps into GTP (N3 interface is not modified).
- In the uplink, traffic is classified at SRGW by UL CL(Uplink Classifier) and steered into an SR policy. The SRGW is a UPF1 functionality, hence it can coexist with UPF UL CL functionality.
- SRGW removes GTP, finds SID list related to DA, add SRH with SID list.

Our reference topology is shown in Figure 6. In this mode we assume that the gNB is an unmodified gNB using IPv4/GTP. The UPFs are SR-aware. Also, as explained before, we introduce a new SRGW entity that is going to map the IPv4/GTP traffic to SRv6.

We also assume that we have two service segment, S1 and C1. S1 represents a VNF in the network, and C1 represents a router over which we are going to perform Traffic Engineering.

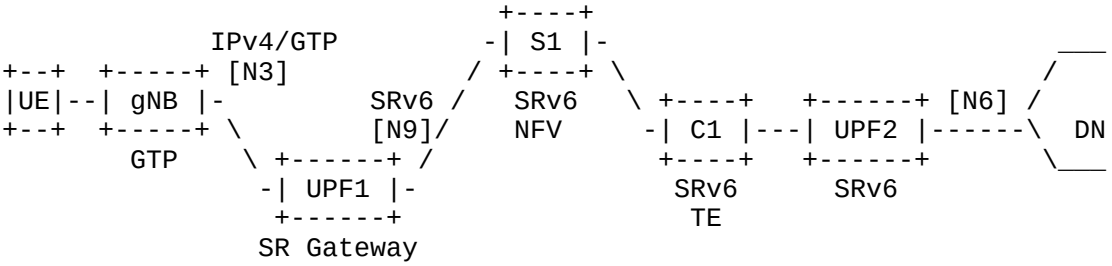


Figure 6: Enhanced mode with unchanged gNB IPv4/GTP behavior

5.3.2.1. Packet flow - Uplink

The uplink packet flow is the following:

```
 gNB_out : (gNB, B)(GTP: TEID T)(A,Z)          -> Interface N3
                                                  unchanged IPv4/GTP
 SRGW_out: (SRGW, S1)(U2::1, C1; SL=2)(A,Z)    -> T.M.Tmap function
 S1_out  : (SRGW, C1)(U2::1, C1; SL=1)(A,Z)
 C1_out  : (SRGW, U2::1) (A,Z)                 -> PSP
 UPF2_out: (A,Z)                               -> End.DT4 or End.DT6
```

The UE sends a packet destined to Z towards the gNB on a specific bearer for that session. The gNB, which is unmodified, encapsulates the packet into a new IPv4, UDP and GTP headers. The IPv4 DA, B, and the GTP TEID are the ones received at the N2 interface.

When the packet arrives to the SRGW -UPF1-, the SRGW has an UL CL (uplink classifier) rule for incoming traffic from the gNB that steers the traffic into an SR policy by using the function T.M.TMap. The SRGW removes the IPv4, UDP and GTP headers and pushes an IPv6 header with its own SRH containing the SIDs related to the SR policy associated with this traffic. The SRGW forwards according to the new IPv6 DA.

The nodes S1 and C1 perform their related Endpoint functionality and forward.

When the packet arrives at UPF2, the active segment is (U2::1) which is bound to End.DT4/6 which performs the decapsulation (removing the outer IPv6 header with all it’s extension headers) and forwards towards the data network.

5.3.2.2. Packet flow - Downlink

The downlink packet flow is the following:

```
UPF2_in : (Z,A)                                -> UPF2 maps flow with SID
                                                  <C1, S1,SRGW::SA:DA:TEID>
UPF2_out: (U2::1, C1)(SRGW::SA:DA:TEID, S1; SL=2)(Z,A) ->T.Encaps.Red
C1_out  : (U2::1, S1)(SRGW::SA:DA:TEID, S1; SL=1)(Z,A)
S1_out  : (U2::1, SRGW::SA:DA:TEID)(Z,A)
SRGW_out: (SA, DA)(GTP: TEID=T)(Z,A)           -> End.M.GTP4.E
gNB_out : (Z,A)
```

When a packet destined to A arrives to the UPF2, the UPF2 performs a lookup in the associated table to A and finds the SID list <C1, S1, SRGW::SA:DA:TEID>. The UPF2 performs a T.Encaps.Reduced operation,

encapsulating the packet into a new IPv6 header with its corresponding SRH.

The nodes C1 and S1 perform their related Endpoint processing.

Once the packet arrives to the SRGW, the SRGW realizes the active SID is an End.M.GTP4.E function. The SRGW removes the IPv6 header and all it's extensions headers. The SRGW generates an IPv4, UDP and GTP headers. The IPv4 SA and DA will the ones received as part of the SID arguments. The TEID in the generated GTP header is also the arguments of the received End.M.GTP4.E SID The SRGW pushes the headers to the packet and forwards the packet towards the gNB.

Once the packet arrives to the gNB, the packet is a regular IPv4/GTP packet. The gNB looks for the specific radio bearer for that TEID and forward it on the bearer. This gNB behavior is not modified from current and previous generations.

5.3.2.3. Scalability

For the downlink traffic, the SRGW is stateless. All the state is in the SRH imposed by the UPF. The UPF must have this UE-base state anyway (it is its anchor point).

For the uplink traffic, the state at the SRGW is dedicated on a per UE/session basis. This is an UL CL (uplink classifier). There is state for steering the different sessions on a SR policies. Notice however that the SR policies are shared among several UE/sessions.

5.3.2.4. IPv6 user-traffic

For IPv6 user-traffic it is RECOMMENDED to perform encapsulation. However based on local policy, a service provider MAY choose to do SRH insertion. The main benefit is a lower overhead.

5.3.3. Extensions to the interworking mechanisms

In this section we presented two mechanisms for interworking with gNBs that do not support SRv6. These mechanism are done to support GTP over IPv4 and GTP over IPv6.

Even though we have presented these methods as an extension to the "Enhanced mode", it is straightforward in its applicability to the "Traditional mode".

Furthermore, although these mechanisms are designed for interworking with legacy RAN at the N3 interface, these methods could also be

applied for interworking with a non-SRv6 capable UPF at the N9 interface (e.g. L3-anchor is SRv6 capable but L2-anchor is not).

6. SRv6 SID Mobility Functions

6.1. End.MAP: Endpoint function with SID mapping

The "Endpoint function with SID mapping" function (End.MAP for short) is used in several scenarios. Particularly in mobility, it is used in the UPFs for the anchor functionality in some of the use-cases.

When a SR node N receives a packet destined to S and S is a local End.MAP SID, N does:

```
1.    look up the IPv6 DA in the mapping table
2.    update the IPv6 DA with the new mapped SID              ;; Ref1
5.    forward according to the new mapped SID
8. ELSE
9.    Drop the packet
```

Ref1: Note that the SID in the SRH is NOT modified.

6.2. End.M.GTP6.D: Endpoint function with decapsulation from IPv6/GTP tunnel

The "Endpoint function with IPv6/GTP decapsulation into SR policy" function (End.M.GTP6.D for short) is used in interworking scenario for the uplink towards from the legacy gNB using IPv6/GTP. This SID is associated with an SR policy <S1, S2, S3> and an IPv6 Source Address A.

When the SR Gateway node N receives a packet destined to S and S is a local End.M.GTP6.D SID, N does:

```
1. IF NH=UDP & UDP_PORT = GTP THEN
2.    pop the IP, UDP and GTP headers
3.    push a new IPv6 header with its own SRH <S2, S3>
4.    set the outer IPv6 SA to A
5.    set the outer IPv6 DA to S1
6.    forward according to the first segment of the SRv6 Policy
7. ELSE
8.    Drop the packet
```

6.3. End.M.GTP6.E: Endpoint function with encapsulation for IPv6/GTP tunnel

The "Endpoint function with encapsulation for IPv6/GTP tunnel" function (End.M.GTP6.E for short) is used in interworking scenario for the downlink towards the legacy gNB using IPv6/GTP.

The End.M.GTP6.E function has a 32-bit argument space. This argument corresponds to the GTP TEID.

When the SR Gateway node N receives a packet destined to S and S is a local End.M.GTP6.E SID, N does:

```
 1. IF NH=SRH & SL = 1  THEN                                ;; Ref1
 2.    decrement SL
 3.    store SRH[SL] in variable new_DA
 4.    store TEID in variable new_TEID                      ;; Ref2
 5.    pop IP header and all it's extension headers
 6.    push new IPv6 header and GTP-U header
 7.    set IPv6 DA to new_DA
 8.    set GTP_TEID to new_TEID
 9.    lookup the new_DA and forward the packet accordingly
10. ELSE
11.    Drop the packet
```

Ref1: An End.M.GTP6.E SID MUST always be the penultimate SID.

Ref2: TEID is extracted from the argument space of the current SID.

6.4. End.M.GTP4.E: Endpoint function with encapsulation for IPv4/GTP tunnel

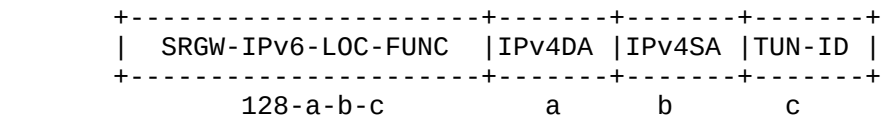
The "Endpoint function with encapsulation for IPv4/GTP tunnel" function (End.M.GTP4.UP for short) is used in the downlink when doing interworking with legacy gNB using IPv4/GTP.

When the SR Gateway node N receives a packet destined to S and S is a local End.M.GTP4.E SID, N does:

```
1. IF NH=SRH & SL > 0 THEN
2.    decrement SL
3.    update the IPv6 DA with SRH[SL]
4.    pop the SRH
4.    push header of TUN-PROTO with tunnel ID from S       ;; Ref1
5.    push outer IPv4 header with SA, DA from S
6. ELSE
7.    Drop the packet
```

Ref1: TUN-PROTO indicates target tunnel type.

Note that S has the following format:



End.M.GTP4.E SID Encoding

6.5. T.M.Tmap: Transit behavior with IPv4/GTP decapsulation and mapping into an SRv6 Policy

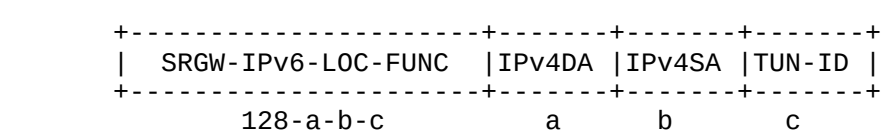
The "Transit with tunnel decapsulation and map to an SRv6 policy" function (T.Tmap for short) is used in the direction from legacy user-plane to SRv6 user-plane network.

When the SR Gateway node N receives a packet destined to a IW-IPv4-Prefix, N does:

```
1. IF P.PLOAD == TUN-PROTO THEN    ;; Ref1
2.    pop the outer IPv4 header and tunnel headers
3.    copy IPv4 DA, SA, TUN-ID to form SID B with SRGW-IPv6-Prefix
4.    encapsulate the packet into a new IPv6 header  ;; Ref2, Ref2bis
5.    set the IPv6 DA = B
6.    forward along the shortest path to B
7. ELSE
8.    Drop the packet
```

Ref1: TUN-PROTO indicates target tunnel type.

Note that B has the following format:

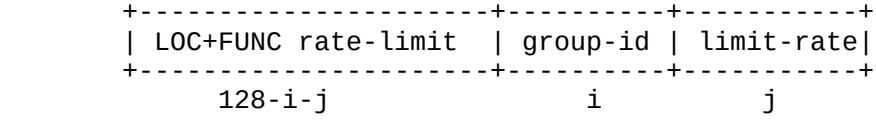


End.M.GTP4.E SID Encoding

Note that the B SID, is going to be an SRv6 BindingSID instantiated at the first UPF (anchor point). A static format is leveraged to instantiate this Binding SIDs in order to remove state from the SRGW.

6.6. End.Limit: Rate Limiting function

Mobile user-plane requires a rate-limit feature. SID is able to encode limiting rate as an argument in SID. Multiple flows of packets should have same group identifier in SID when those flows are in an same AMBR group. This helps to keep user-plane stateless. That enables SRv6 endpoint nodes which are unaware from the mobile control-plane information. Encoding format of rate limit segment SID is following:



End.Limit: Rate limiting function argument format

In case of j bit length is zero in SID, the node should not do rate limiting unless static configuration or control-plane sets the limit rate associated to the SID.

7. Network Slicing Considerations

A mobile network may be required to implement "network slices", which logically separate network resources. User-plane functions represented as SRv6 segments would be part of a slice.

A simple way to represent slice would be to apply L2/L3 VPN described in [I-D.filsfils-spring-srv6-network-programming]. Segment Routing with [I-D.hegdeppsenak-isis-sr-flex-algo] provides even more advanced separation based on metrics like link-delay. Thus, a service provider would be able to have network slices per required SLA.

The SRv6 SID and quite a few SR extended capability would be a powerful tool for providing logical separation/integration within a network. Details are for further study.

8. Control Plane Considerations

This documents focuses on the dataplane behavior. The control planes could be based on the existing 3GPP based signalling for N4 interface [TS.29244], [I-D.ietf-dmm-fpc-cpdp], control-plane protocols described in [WHITEPAPER-5G-UP], etc. and to be discussed further.

Note that the IANA section of this document allocates the SRv6 endpoint function types for the new functions defined in this document. All control-plane protocols are expected to leverage these function type-codes to signal each function.

It’s notable that SRv6’s network programming nature allows a flexible and dynamic anchor placement.

9. Security Considerations

TBD

10. IANA Considerations

This I-D requests to IANA to allocate, within the "SRv6 Endpoint Types" sub-registry belonging to the top-level "Segment-routing with IPv6 dataplane (SRv6) Parameters" registry [I-D.filsfils-spring-srv6-network-programming], the following allocations:

Value/Range	Hex	Endpoint function	Reference
TBA	TBA	End.MAP	[This.ID]
TBA	TBA	End.M.GTP6.D	[This.ID]
TBA	TBA	End.M.GTP6.E	[This.ID]
TBA	TBA	End.M.GTP4.E	[This.ID]
TBA	TBA	End.Limit	[This.ID]

Table 1: SRv6 Mobile User-plane Endpoint Types

11. Acknowledgements

The authors would like to thank Daisuke Yokota, Bart Peirens, Ryokichi Onishi, Kentaro Ebisawa, Peter Bosch and Darren Dukes for their useful comments of this work.

12. References

12.1. Normative References

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Leddy, J., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R., Matsushima, S., Lebrun, D., Decraene, B., Peirens, B., Salsano, S., Naik, G., Elmalky, H., Jonnalagadda, P., Sharif, M., Ayyangar, A., Mynam, S., Henderickx, W., Bashandy, A., Raza, K., Dukes, D., Clad, F., and P. Camarillo, "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming-03 (work in progress), December 2017.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-15 (work in progress), January 2018.

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

12.2. Informative References

[I-D.gundavelli-dmm-mfa]
Gundavelli, S., Liebsch, M., and S. Matsushima, "Mobility-aware Floating Anchor (MFA)", draft-gundavelli-dmm-mfa-00 (work in progress), February 2018.

[I-D.hegdeppsenak-isis-sr-flex-algo]
Psenak, P., Hegde, S., Filsfils, C., and A. Gulko, "ISIS Segment Routing Flexible Algorithm", draft-hegdeppsenak-isis-sr-flex-algo-02 (work in progress), February 2018.

[I-D.ietf-dmm-fpc-cpdp]
Matsushima, S., Bertz, L., Liebsch, M., Gundavelli, S., Moses, D., and C. Perkins, "Protocol for Forwarding Policy Configuration (FPC) in DMM", draft-ietf-dmm-fpc-cpdp-09 (work in progress), October 2017.

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

[TS.23501]
3GPP, , "System Architecture for the 5G System", 3GPP TS 23.501 15.0.0, November 2017.

[TS.29244]
3GPP, , "Interface between the Control Plane and the User Plane Nodes", 3GPP TS 29.244 15.0.0, December 2017.

[TS.29281]
3GPP, , "General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)", 3GPP TS 29.281 15.1.0, December 2017.

Authors' Addresses

Satoru Matsushima
SoftBank
Tokyo
Japan

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

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

Miya Kohno
Cisco Systems, Inc.
Japan

Email: mkohno@cisco.com

Pablo Camarillo Garvia
Cisco Systems, Inc.
Spain

Email: pcamaril@cisco.com

Daniel Voyer
Bell Canada
Canada

Email: daniel.voyer@bell.ca

Charles E. Perkins
Futurewei Inc.
2330 Central Expressway
Santa Clara, CA 95050
USA

Phone: +1-408-330-4586
Email: charliep@computer.org