Abstract

Current standardization effort on the evolution of the mobile communication system reconsiders the mobile data plane protocol. The IETF DMM Working Group has work that proposes and analyzes various protocols as alternative to the GPRS Tunneling Protocol for User Plane (GTP-U) for an overlay deployment in between the mobile device’s assigned data plane anchor and its current radio base station, which are denoted as N9 and N3 interfaces. In the view of some future deployment and the original intent per the very early DMM WG charter, a mobile device’s data plane anchor may be highly distributed and re-selected for optimization throughout a mobile device’s communication with one or more correspondent services. Such re-configuration has impact on the packet routing in between the mobile device’s data plane anchor and the one or multiple data networks hosting the services, which is denoted as N6 interface. This draft proposes and discusses a solution to control, setup and maintain traffic treatment policy on the cellular communication system’s N6 interface while taking the UE’s PDU session settings per the cellular system’s control plane, such as QoS and locator information, into account.

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 12, 2019.
1. Terminology

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

2. Introduction

Recent releases and deployments of cellular mobile communication systems utilize an overlay on the mobile data plane to forward a mobile device’s data packets in between the mobile device and an anchor point, which serves as first hop router to the mobile device. The overlay is realized by the GPRS Tunneling Protocol for user plane
(GTP-U), which is able to carry network-specific attributes in the
tunnel protocol headers.

The 3rd Generation Partnership Project (3GPP) is in charge of the
cellular mobile communication system’s specification and is currently
finalizing a 15th release, which has fundamental changes compared to
previous releases. Such changes include a clean split between
control- and data plane functions, more flexible deployment and re-
configuration of data plane anchors, as well as support for local
data network (DN) access and multi-homing.

In between a mobile device’s current radio base station in the radio
access network (RAN) and its data plane anchor, the release 15
specification assumes an overlay per the previous releases utilizing
GTP-U. The data plane anchor is denoted as User Plane Function (UPF)
to anchor a Packet Data Unit (PDU) Session for the mobile device.
This draft abbreviates the UPF, which serves a device’s PDU session
anchor, as UPF_a. In between a UPF_a and the device’s current radio
base station, none, one or multiple additional UPFs can be deployed
to classify uplink traffic in support of policy-based routing to a
particular DN without traversing the UPF_a. This draft denotes such
intermediate UPF as UPF_i. Interfaces between a DN and a mobile
device’s UPF_a is denoted as N6, the interface between a UPF_i and
one or multiple UPF_a is denoted as N9, and the interface between a
UPF_i and a radio base station is denoted as N3. Whereas regular
routing of mobile devices’ PDUs is assumed on N6, N9 and N3 deploy a
GTP-U overlay with UPF_a, UPF_i and the radio base station serving as
tunnel endpoints. This end-to-end architecture is depicted in
Figure 1. For a more detailed description of anchor and intermediate
UPF and associated deployment and operation, please refer to
[I-D.bogineni-dmm-optimized-mobile-user-plane] and the 3GPP
specification [TS23.501].

```
+-----+  N3    +-------+  N9    +-------+  N6  /  data  
device---+ RAN +======/==+ UPF_i +=====/====+ UPF_a +-----/--+ network |
+-----+  GTP-U  +-------+  GTP-U   +-------+    IP   
     tunnel            tunnel              PDUs
```

Figure 1: Architecture and interfaces of a 3GPP release 15 data plane
in between a data network and a mobile device.

In alignment with the 3GPP’s current directions to study data plane
protocol candidates which can serve as suitable alternative to GTP-U,
the IETF’s DMM WG has valuable ongoing individual work that analyzes
the GTP-U protocol and derives requirements for an alternative mobile
data plane protocol [I-D.hmm-dmm-5g-uplane-analysis], as well as work
that investigates the use of alternative protocol candidates based on SRv6, ID-Locator separation, and locator re-writing in the current release 15 system architecture [I-D.bogineni-dmm-optimized-mobile-user-plane]. The focus of these drafts is on N9 and N3.

In the view of optimization options on the complete end-to-end data plane, [I-D.gundavelli-dmm-mfa] complements other draft and proposes data plane optimization on N6. Such operation is of particular interest when the mobile device’s UPF_a is decentralized and deployed close to the device’s current radio base station. Such deployment may be preferable for some services, such as edge computing and access to associated edge DNs, and mitigates the role of the UPF_i and N9 interfaces. In particular the selection and configuration of UPF_i instances can omitted and associated signaling costs can be saved. However, such deployment strengthens the expectation on IP-based PDU routing on N6, as the serving DN may not be always topologically close to the device and its current UPF_a. Such requirements include QoS support on N6, metering support and traffic steering in case the mobile device’s UPF_a changes while its IP address and associated sessions should continue.

The same requirements on N6 apply for multi-homing per [TS23.501] where the mobile device’s UPF_a is close to a first DN (DN1) whereas a UPF_i is used to enable access to a second DN (DN2), either through a secondary UPF_a close to DN2 or directly from the UPF_i, without the use of a secondary UPF_a. Since services in both DNs address the same IP address of the mobile device (IP_ue) to send downlink traffic, both DNs’ traffic need to be forwarded to the most suitable (e.g. closest) UPF_a or UPF_i respectively.

This draft focuses on a solution to control, setup and maintain such dedicated routes and additional traffic treatment policy on N6, while taking the UE’s PDU session settings per the cellular system’s control plane, such as QoS and locator information, into account.

3. Positioning of N6 policy control

This section briefly introduces the relevant mobile system architecture components and interfaces, and covers some high-level use cases which can benefit from data plane policy control on N6 interface endpoints.

3.1. System architecture for mobile access to data networks

The 3GPP’s 5G system architecture introduces in the core network a clear control-/user plane separation (CUPS), in order to have flexible deployment of the different functions (e.g., user plane
nodes can scale independently from control plane elements in case of user traffic growth). Again to leverage flexibility and efficiency, the control plane is split in different functions, each offering a specific service, in the so called Service Based Architecture (SBA).

Among all the control plane functions, the Session Management function (SMF) takes care of the session management (session establishment, modification, release), IP allocation and selection of an IP anchor point for the session, as well as traffic steering in between UPFs and radio base stations. In order to manage the user session, the SMF collaborates with other control plane services (e.g., Policy Control Function - PCF - providing policy rules for traffic treatment and monitoring), in particular with the Access and Mobility Management Function (AMF), which manages registration, authentication and authorization and security context. One of the main task of the SMF is to instruct User Plane Functions (UPFs), through N4 interface. When a new session is to be created, the SMF selects one or multiple UPFs for the user traffic and selects one UPF as session anchor (UPF_a). UPF_a acts as a proxy for user traffic, which means all traffic directed to the UE passes through the UPF anchor. Beside the UPF_a, if other UPFs are present (i.e., between the radio base station and the UPF_a), these are deployed as classifiers for user uplink traffic.

In Figure 2 a simplified 5G architecture [TS23.501] is depicted, showing two Data Networks (DN) to whom a user may need a connection. To each Data Network a UPF_a is associated, acting as session anchor and providing to the user an IP address needed for the connection. UPF_a also acts as tunnel termination point, since user traffic is encapsulated on both N3 and N9 interfaces, using the GPRS Tunneling Protocol for User Plane (GTP-U). Whereas, on N6 interface IP PDUs are routed without tunneling.
communication between control plane functions

```
+--+--
| AMF | SMF |
+-----+-----+
| N2  | N4   |
```

```
/ N4 | N4
+/-----+
```

```
/ N2                N4|   |N4
```

```
/               +------+   +------+
```

```
/               |                 |            _________
```

```
+----+     +-----+   N3    +---+---+   N9    +---+---+    N6 /  data   \
```

```
| UE |-----+ RAN +=========+ UPF_i +=========+ UPF_a +----/--+ network |
```

```
| IP_ue proxy| IP_ue proxy |
```

```
| IP_ue+-------+ proxy IP_ue |
```

```
| N6 /  data   |
```

```
+-------/----+ network |
```

```
| IP \____1____/   |
```

```
| proxy IP_ue |
```

```
| IP \____2____/   |
```

Figure 2: Data plane with a simplified release 15 control plane

Data networks host Application Servers (AS), which provide services to UEs, and an internal network comprising data plane nodes (DPN), such as routers and switches, to connect the services with the transport network. Both, the transport network and the data network’s internal network build the N6 interface, which is depicted in Figure 3. In order to apply traffic treatment policy to uplink traffic in between a UPF and a data network, the UPF receives policies via the N4 interface. For downlink traffic, the AS/DPN should have means to receive traffic treatment policies.

A way to enforce N6 policies to the DPN/AS in a data network is needed. It is evident that this rule must originate from the cellular control plane due to its knowledge about the UE’s states, such as its locator or QoS, and when these states are updated or re-configured. Different means to convey and enforce associated traffic treatment policies in a DPN/AS exist, such as the use of routing protocols or control-/data plane configuration protocols.
3.2. Use cases with demand for N6 traffic treatment policy

The motivations behind the need for N6 treatment policy are many. Following, some of the use cases are listed and described.

UE to UE communication: a scenario which is not explicitly shown in Figure 2 and Figure 3 is UE to UE communication, when a UPF_a via N6 interface is connected to another UPF_a (belonging to the same or to another network, and controlled by the same or another SMF), with the latter UPF being associated to a second UE. In this scenario, all the data plane elements on the path are controlled by control plane elements of the 5GC (i.e., SMFs), but anyway additional policies on N6 may be forwarded in order to steer traffic on an optimized route directly towards the edge UPF for the specific UE, without passing through the UPF_a.

UE to edge data network: in this use case, the UE connects to an edge Data Network, meaning a DN positioned at the edge of the core network, near to the access network (typical MEC scenario). In mobility, a new UPF_a may be assigned to UE, and routes to the previous edge network would follow a non-optimized path, passing through the new UPF_a for the UE. With traffic treatment policies...
this can be avoid, giving a traffic steering policy to the DPN in charge for the edge DN.

Concurrent use of multiple data networks: a possible scenario is the one in which a UE collects the desired content from different data networks (e.g., because of Content Delivery Networks - CDN). To optimize routing in this scenario, the downlink traffic should traverse for each data network the optimized path through the UE and not be forced through a (central) UPF_a common to all the data networks. Again, this can be done with policies on N6 interface. This particular use case also highlights the importance to consider optimization on N6, whereas other works focus on N9: considering a UPF_a near the data network, as proposed in other solutions, would not allow multiple DN access in an unique user session and so would not allow for content access on different destinations.

4. N6 traffic treatment - Requirements and policy types

Use cases for traffic treatment on N6 per a data plane policy include cases where the UPF_a is deployed closer at the mobile edge, e.g. to not only access a local data network in the proximity of the UE, but also other data networks sharing the single edge UPF_a. In that case the N6 interface may span some distance in the transport network in between the data network(s) and the UPF_a. Dependent on the expected QoE/QoS of the traffic, traffic treatment policies for QoS differentiation, packet labeling, etc. may apply to the UE’s packets on N6. For uplink traffic, the UE’s UPF_a can enforce such traffic treatment policies to uplink traffic, where a DPN associated with the data network(s) (e.g. PE router, transit router, router/switch of the data center transport network, TOR switches of Application Servers, etc.) enforces such policies to downlink traffic.

The same need for traffic treatment policies applies to traffic between a UPF_i, which classifies uplink traffic for forwarding to a local data network, and the data network. Downlink traffic from the local data network to the UE should then be forwarded towards the UPF_i, not via the UE’s UPF_a.

In advanced scenarios, the SMF may decide to reconfigure the UE’s UPFs, e.g. by relocating the UPF_a or a UPF_i while maintaining the UE’s IP address (IP_ue) and data sessions using this IP address. In such case, a DPN associated with the one or multiple data networks, which run correspondent services for the UE, must enforce traffic steering policies to downlink traffic to achieve routing of downlink traffic to the UE’s current UPF_a or UPF_i respectively.

In summary, traffic treatment policies that apply to a UE’s uplink and downlink traffic on N6 include the following types:
Requirements for N6 traffic treatment include the following:

- Awareness of UE location information (first hop router accuracy, UPF_a/UPF_i) - Set or update DPN policy for traffic steering
- Awareness of topology - Select and update most suitable UPF (UPF_a/UPF_i) for the communication with a data network, e.g. after UPF changed
- Availability of initial or updated policies when needed
- No/Low impact on data traffic (packet loss, re-ordering) when policies are updated - DPNs may request/solicit policies or get notified about initial and updated policies

5. Leveraging the mobile control plane for N6 policy control

Methods for N6 policy control consist in instructing the DPNs with rules for traffic steering, QoS policies enforcing, etc. The solution described in this draft is based on leveraging the mobile control plane, in order to introduce some logic to manage and forward policies to DPNs on N6 interface. To do this, the Application Function (AF) defined in 5GS [TS23.501] is used as binding element between the cellular network control plane and the data network data plane.

Per [TS23.501], the AF is introduced to inter-work with the Policy Control Function (PCF) in order to condition and contribute to some SMF decisions. This happens with the AF sending specific requests to the PCF and the latter translating those requests in policies for the SMF. Depending on the domain in which the AF is located, a Network Exposure Function (NEF) may be in between to enable the AF collaborating with the other control plane elements of the cellular architecture.

In support of the proposed scenario, the AF can solicit data plane policies from the cellular control plane by sending a request. At reception of the policies, the AF can pass the policies on for
further processing and enforcement in the data network’s AS/DPN. In this way, DPNs receive from the control plane policies for the user traffic traversing them. The AF may be co-located with a control function, which utilizes the DMM WG’s Forwarding Policy Configuration (FPC) protocol to implement policies in the AS/DPN, or leverage an SDN controller for the selection and configuration of AS/DPN.

The policies defined and forwarded by the AF are based on the status of the mobile network, which the AF can obtain from the SMF. In any moment, in fact, the SMF is in charge of keeping track of the selected UPFs and of monitoring the user session. Based on this information, the AF forwards specific rules to a DPN (e.g., traffic steering rules to make the user’s traffic reach the most suitable UPF_a). In some cases (e.g., user mobility), the SMF can also change UPFs for a specific user and in this case the AF will receive updated policies for enforcement in the involved AS/DPN.

Figure 4 shows how the previous architecture evolves with the introduction of the AF.

![Diagram showing the communication between control plane functions](image)
6. N6 endpoints - loose and tight coupling options

As described in the previous section, we take advantage of the Application Function (AF) to bind the 3GPP’s domain functions with those introduced in this draft for N6 policy enforcement. According to [TS23.501], an Application Function may send requests to influence SMF decisions for User Plane (UP) traffic of PDU Sessions (e.g., based on the relocation of an application on the Data Network side, the AF can notify this to the SMF in order to trigger a relocation of UPF(s) from the SMF, to choose a new UPF more suitable for the new Data Network).

In addition, the AF can subscribe to events from the SMF in order to receive notification about UP management events (e.g., when a PDU Session anchor has been established or released).

As defined in [TS23.502], the AF interacts with the PCF/SMF via the NEF or directly and the PCF then forwards requests from the AF towards the SMF as Session Management (SM) Policies. For the sake of simplicity, in this section all the 3GPP’s functions apart from the AF are collected under the name of "3GPP’s C-PLANE", and the specific service to which the AF interacts in the 3GPP C-PLANE is not relevant for this draft.

In order to forward specific policies to the Data Plane Nodes/ Application Servers (DPNs/ASs) associated with each Data Network, a Network Controller (NC) is considered to be co-located with the AF element. The NC performs the selection of a DPN/AS element based on the received information from the C-PLANE. The AF/NC forwards control messages to a DPN/AS through an AFNC-CPUP interface, giving indications to steer the downlink traffic properly and coherently with the UP updates from the 3GPP’s side.

Forwarding N6 polices to the N6 endpoints involved (i.e., UPF and DPN) can happen in two different ways:

1) Tight coupling scenario: The UPF can enforce policies per the AF/NC decisions. The UPF receives associated policies from the 3GPP’s C-PLANE. The corresponding DPN/AS receives the policy via the AFNC-CPUP interface.

2) Loose coupling scenario: A separate DPN function is co-located with the UPF. Main policies for N6 traffic treatment do not traverse the 3GPP’s C-PLANE but are controlled at both N6 interface endpoints’ DPN by the AF/NC via the AFNC-CPUP interface.
In the tight coupling scenario, the N6 interface configurations for the UPF are all enforced though the 3GPP domain. Therefore, the 3GPP’s C-PLANE interacts with the AF/NC element through the AFNC_3GPP interface and receives on this interface requests to influence the UP traffic policies. 3GPP decides if enforce those policies on the UPF(s) involved.

The architecture and interfaces involved in this tight coupling scenario are depicted in Figure 5.

Figure 5: N6 endpoints tight coupling scenario

In Section 7.1 the operation flow and information model for the messages exchanged in this type of coupling are presented and described. Both the cases of a AF/NC-initiated and 3GPP-initiated message flow are considered.

In the loose coupling scenario, an additional DPN element is associated with a UPF and represents a key element to enforce N6 traffic treatment policies on the UPF-side of the N6 interface. This DPN is controlled by the AF/NC through the AFNC_CPUP interface, as depicted in Figure 6.

Loose coupling allows reducing 3GPP’s role in the N6 endpoint management, potentially allowing under certain assumptions (e.g., no UPF re-selection is needed), an optimized control of the N6 interface from the AF/NC element, transparently from 3GPP’s domain. This kind
of scenario results as an advantage particularly for use cases in which the UPF is deployed in the proximity of the Data Network and far from the 3GPP’s C-PLANE (i.e., in a Mobile Edge Computing - MEC - alike scenario).

For particular cases which request 3GPP’s C-PLANE involvement (i.e., UPF re-selection or other changes not related to the only N6 endpoint) the AFNC_3GPP is still used for notifications and requests between the AF/NC and the 3GPP’s C-PLANE.

![Diagram of N6 endpoints loose coupling scenario](image)

Figure 6: N6 endpoints loose coupling scenario

7. Operations for N6 policy enforcement in a tight coupling scenario

In the following sub-sections, message sequences are shown assuming a tight coupling scenario between N6 interface endpoints, as depicted in Figure 5. Two different operation flows can be distinguished, based on the entity initiating and requesting for the N6 policy. Section 7.1 describes the message sequence in the case of AF/NC-initiated N6 policy request, while Section 7.2 covers the alternative case in which the request for a N6 policy is initiated from the 3GPP’s C-PLANE.

In the message sequences, special attention is given to the AFNC_CPUP and AFNC_3GPP interfaces defined in this draft and Information Models for messages exchanged on those interfaces are provided.
7.1. AF/NC-initiated N6 policy enforcement

A N6 policy can be triggered from the AF/NC element and is then forwarded directly to the DPN N6 endpoint (through AFNC_CPUP interface) and indirectly to the UPF N6 endpoint (through AFNC_3GPP interface).

As example, the AF/NC may request updated n6 policies for the following reasons:

- there is the need of a different QoS to be applied to traffic, which is identified in the request.
- there is the need for a re-location of the application to a different Data Network and therefore changes for traffic in uplink on the UPF’s N6 endpoint should be applied.

Figure 7 depicts the AF/NC-initiaed N6 policy enforcement message sequence.

```
+--------+     +--------+ |   +-------+    +--------+
|C-PLANE|     | UPF(s) |-+   | AF/NC |    | DPN/AS |
+---+---+     +--+-----+     +---+---+    +---+----+
|            |               |            |
|            |               |            |
|<----(1)TRAFFIC------------|            |
|    INFLUENCE REQUEST       |            |
|            |               |            |
|---(2a)---->|               |            |
|<----(2b)---|               |            |
|  N4 SESSION EST/MOD/REL    |            |
|            |               |            |
|------(3)TRAFFIC----------->|            |
|  INFLUENCE RESPONSE        |            |
|            |               |---(4a)---->|
|            |               |<---(4b)----|
|            |               |  AFNC_CPUP |
|            |               | CONFIGURATION|
```

Figure 7: Message flow for AF/NC-initiated N6 policy enforcement

Following, a description for each message is given:
TRAFFIC INFLUENCE REQUEST: this message is sent from the AF/NC to the 3GPP’s C-PLANE in order to request a modification for UP traffic. The message contains the fields listed in Table 1.

<table>
<thead>
<tr>
<th>Message Fields</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request ID</td>
<td>Identifies the current request in order to match it with following response messages.</td>
<td></td>
</tr>
<tr>
<td>Traffic Identifier</td>
<td>Identifies the UP traffic which is targeted by the request. Traffic may be identified based on the session, UE-based or even slice-based (i.e., addressing all the traffic belonging to a specific network slice).</td>
<td>3GPP’s identifiers defined in [TS23.501] may be used to identify traffic (e.g., DNN for traffic toward a specific Data Network, NSSAI for a specific slice, UE GUTI for a specific user, etc.)</td>
</tr>
<tr>
<td>QoS parameters</td>
<td>Contains the QoS parameters for the targeted traffic</td>
<td></td>
</tr>
<tr>
<td>DPN N6 endpoint</td>
<td>Brings information about the N6 endpoint on the Data Network side.</td>
<td></td>
</tr>
</tbody>
</table>

Table 1

Based on the N6 endpoint information, the 3GPP’s C-PLANE may take decisions on UPF(s) selection and re-location. For instance, this field could carry a Data Network Access ID (DNAI), identifying a specific Data Network on which the 3GPP’s domain could select the best matching UPF (e.g., based on proximity).

N4 SESSION ESTABLISHMENT/MODIFICATION/RELEASE: this are 3GPP’s messages defined in [TS23.502] and used to enforce changing to one or more UPF or to select and configure a new UPF. Through this messages, the N6 policies requested from the AF/NC can be enforced to the UPF(s).
(3) TRAFFIC INFLUENCE RESPONSE: this message is sent from the 3GPP’s C-PLANE to the AF/NC in order to acknowledge the UP changes made based on the previous request message. The message contains the fields listed in Table 2.

<table>
<thead>
<tr>
<th>Message Fields</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request ID</td>
<td>Identifies the request message to which this response is referred to.</td>
<td>-</td>
</tr>
<tr>
<td>Traffic Identifier</td>
<td>Identifies the UP traffic which is targeted by the request. Traffic may be identified based on the session, UE-based or even slice-based (i.e., addressing all the traffic belonging to a specific network slice).</td>
<td>Traffic actually influenced could differ from the original traffic targeted in the request.</td>
</tr>
<tr>
<td>UPF N6 endpoint</td>
<td>Brings information about the N6 endpoint on the 3GPP’s side.</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2

N6 endpoint information on 3GPP’s side (e.g., IP address of the N6 endpoint UPF) are used from the AF/NC to set the DPN(s) in order to properly route downlink traffic.

(4a)(4b) AFNC_CPUP CONFIGURATION: This message is used to instruct the DPN(s) involved in the UP changes. For instance, in case of UPF re-selection and UPF’s N6 endpoint (e.g., IP address) changing, traffic steering rules for downlink traffic need to be enforced to the DPN. The structure of this message is out of the scope of this draft and candidates for managing this interface are already present (e.g., Forwarding Policy Configuration (FPC) defined in [FPC]).

7.2. 3GPP-initiated N6 policy enforcement

A N6 policy can be triggered by the 3GPP domain. In this case, an initial subscription mechanism is needed, in which one or multiple AF subscribe the 3GPP’s C-PLANE in order to receive notification about the subscribed events. Some of the events, of which a AF/NC could be interested in, are:
o re-selection one or multiple UPF(s) from the 3GPP’s C-PLANE.

o changes in the UP traffic QoS parameters.

o etc.

Figure 8 depicts the message sequence described the AF subscription and a notification from the 3GPP’s domain when the specific event occurs.

```
+--------+     +--------+ |   +-------+    +--------+
|C-PLANE|     | UPF(s) |-+   | AF/NC |    | DPN/AS |
+---+---+     +--+-----+     +---+---+    +---+----+
|            |               |            |
|<-----(0)EVENT--------------|            |
|    SUBSCRIPTION            |            |
|___|          |               |            |
|event          |               |            |
|------(1)EVENT ------------>|            |
|      NOTIFICATION          |            |
|            |               |----(2a)--->|
|            |               |<---(2b)----|
|            |               |  AFNC_CPUP |
|            |               |   CONFIGURATION |
```

Figure 8: Message flow for 3GPP-initiated N6 policy enforcement

The messages used are here described:

(0) EVENT SUBSCRIPTION: this message is sent from the AF/NC to the 3GPP’s C-PLANE in order for the AF/NC to subscribe to some specific UP events. When received from the 3GPP’s C-PLANE, all future UP events (e.g., UPF re-selection, changing in UP traffic parameters) which match with the subscription will be notified to the AF/NC. This message fields are listed in Table 3.
Information model for EVENT SUBSCRIPTION message

<table>
<thead>
<tr>
<th>Message Fields</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscription ID</td>
<td>Identifies the subscription in order to then match the resulting notification.</td>
<td>-</td>
</tr>
<tr>
<td>Event</td>
<td>Identifies the type of event to which the subscription is referred. For instance, the subscription could refer only to an UPF re-selection event, or may refer to any event for the targeted traffic.</td>
<td>Can be 'all-events' or identify a specific type of event.</td>
</tr>
<tr>
<td>Traffic Identifier</td>
<td>Identifies the UP traffic which is targeted by the request. Traffic may be identified based on the session, UE-based or even slice-based (i.e., addressing all the traffic belonging to a specific network slice).</td>
<td>3GPP’s identifiers defined in [TS23.501] may be used to identify traffic (e.g., DNN for traffic toward a specific Data Network, NSSAI for a specific slice, UE IP address for a specific user, etc.).</td>
</tr>
</tbody>
</table>

Table 3

(1) EVENT NOTIFICATION: this message is sent from the 3GPP’s C-PLANE to the AF/NC, triggered by the subscribed event for the targeted traffic. If no subscription for the specific traffic and event was received before the modification occurs the 3GPP’s C-PLANE will not provide any notification for the UP traffic changes. Table 4 lists the field contained in the message.
Information model for EVENT NOTIFICATION message

<table>
<thead>
<tr>
<th>Message Fields</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscription ID</td>
<td>Identifies the subscription message to which this notification is referred to.</td>
<td>-</td>
</tr>
<tr>
<td>Traffic Identifier</td>
<td>Identifies the UP traffic which has been change.</td>
<td>Even if there is no notification for traffic which has not been targeted through a subscription, this field may refer to a subset of the traffic targeted in the subscription (e.g., subscription to a specific user traffic and modification of only one PDU sessions for that user).</td>
</tr>
<tr>
<td>QoS parameters</td>
<td>Brings information about QoS parameters which have been changed.</td>
<td>-</td>
</tr>
<tr>
<td>UPF N6 endpoint</td>
<td>Brings information about the N6 endpoint on the 3GPP’s side which have been changed.</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4

(2a)(2b) AFNC_CPUP CONFIGURATION: This message is used to instruct the DPN(s) involved in the UP changes. For instance, in case of UPF re-selection and UPF’s N6 endpoint (e.g., IP address) changing,
traffic steering rules for downlink traffic need to be enforced to the DPN. The structure of this message is anyway out of the scope of this draft and candidates for managing this interface are already present (e.g., Forwarding Policy Configuration (FPC) defined in [FPC]).

8. IANA Considerations

No IANA action is required for this version of the draft.

9. Security Considerations

Since the solution proposed in this document utilizes the AF to solicit and receive N6 traffic treatment policies from the cellular system’s control plane, the trust relationship between the AF and the cellular system’s domain matters. In case the AF is located in a different administrative domain, the communication from and to the AF may happen via the system’s Network Exposure Functions (NEF). The semantic to request and receive the N6 policy at the AF and in particular the policy types and their descriptions must be aligned to the trust relationship.

Also, the trust relationship between the AF and the DPN/AS matters and a secure direct or indirect (e.g. through an Network Controller) interface, must be ensured.

10. Acknowledgments

The research leading to these results has been partially supported by the H2020-MSCA-ITN-2016 framework under grant agreement number 722788 (SPOTLIGHT).

Authors want to thank Sri Gundavelli, John Kaippallimalil and Shunsuke Homma for their interest and feedback to the use cases and the solution principles for N6 traffic treatment policies.

11. Normative References


Authors’ Addresses

Umberto Fattore
NEC Laboratories Europe GmbH
Kurfuersten-Anlage 36
D-69115 Heidelberg
Germany

Email: umberto.fattore@neclab.eu

Marco Liebsch
NEC Laboratories Europe GmbH
Kurfuersten-Anlage 36
D-69115 Heidelberg
Germany

Email: marco.liebsch@neclab.eu
Distributed Mobility Anchoring
draft-ietf-dmm-distributed-mobility-anchoring-13

Abstract

This document defines distributed mobility anchoring in terms of the different configurations and functions to provide IP mobility support. A network may be configured with distributed mobility anchoring functions for both network-based or host-based mobility support according to the needs of mobility support. In a distributed mobility anchoring environment, multiple anchors are available for mid-session switching of an IP prefix anchor. To start a new flow or to handle a flow not requiring IP session continuity as a mobile node moves to a new network, the flow can be started or re-started using an IP address configured from the new IP prefix anchored to the new network. If the flow needs to survive the change of network, there are solutions that can be used to enable IP address mobility. This document describes different anchoring approaches, depending on the IP mobility needs, and how this IP address mobility is handled by the network.

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."
1. Introduction

A key requirement in distributed mobility management [RFC7333] is to enable traffic to avoid traversing a single mobility anchor far from an optimal route. This document defines different configurations, functional operations and parameters for distributed mobility anchoring and explains how to use them to avoid unnecessarily long routes when a mobile node moves.
Companion distributed mobility management documents are already addressing the architecture and deployment
[I-D.ietf-dmm-deployment-models], source address selection
[I-D.ietf-dmm-ondemand-mobility], and control-plane data-plane
signaling [I-D.ietf-dmm-fpc-cpdp]. A number of distributed mobility
solutions have also been proposed, for example, in
[I-D.seite-dmm-dma], [I-D.ietf-dmm-pmipv6-dlif],
[I-D.sarikaya-dmm-for-wifi], [I-D.yhkim-dmm-enhanced-anchoring], and
[I-D.matsushima-stateless-uplane-vepc].

Distributed mobility anchoring employs multiple anchors in the data
plane. In general, control plane functions may be separated from
data plane functions and be centralized but may also be co-located
with the data plane functions at the distributed anchors. Different
configurations of distributed mobility anchoring are described in
Section 3.1.

As a Mobile Node (MN) attaches to an access router and establishes a
link between them, a /64 IPv6 prefix anchored to the router may be
assigned to the link for exclusive use by the MN [RFC6459]. The MN
may then configure a global IPv6 address from this prefix and use it
as the source IP address in a flow to communicate with its
correspondent node (CN). When there are multiple mobility anchors
assigned to the same MN, an address selection for a given flow is
first required before the flow is initiated. Using an anchor in a
MN’s network of attachment has the advantage that the packets can
simply be forwarded according to the forwarding table. However,
after the flow has been initiated, the MN may later move to another
network which assigns a new mobility anchor to the MN. Since the new
anchor is located in a different network, the MN’s assigned prefix
does not belong to the network where the MN is currently attached.

When the MN wants to continue using its assigned prefix to complete
ongoing data sessions after it has moved to a new network, the
network needs to provide support for the MN’s IP address -- and
session continuity, since routing packets to the MN through the new
network deviates from applying default routes. The IP session
continuity needs of a flow (application) determines how the IP
address used by this flow has to be anchored. If the ongoing IP flow
can cope with an IP prefix/address change, the flow can be
reinitiated with a new IP address anchored in the new network. On
the other hand, if the ongoing IP flow cannot cope with such change,
mobility support is needed. A network supporting a mix of flows both
requiring and not requiring IP mobility support will need to
distinguish these flows.
2. Conventions and Terminology

All general mobility-related terms and their acronyms used in this document are to be interpreted as defined in the Mobile IPv6 (MIPv6) base specification [RFC6275], the Proxy Mobile IPv6 (PMIPv6) specification [RFC5213], the "Mobility Related Terminologies" [RFC3753], and the DMM current practices and gap analysis [RFC7429]. These include terms such as mobile node (MN), correspondent node (CN), home agent (HA), home address (HoA), care-of-address (CoA), local mobility anchor (LMA), and mobile access gateway (MAG).

In addition, this document uses the following terms:

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

Anchoring (of an IP prefix/address): An IP prefix, i.e., Home Network Prefix (HNP), or address, i.e., HoA, assigned for use by an MN is topologically anchored to an anchor node when the anchor node is able to advertise a connected route into the routing infrastructure for the assigned IP prefix. The traffic using the assigned IP address/prefix must traverse the anchor node. We can refer to the function performed by IP anchor node as anchoring, which is a data plane function.

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

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

In a client-server protocol model, location query and update messages may be exchanged between a Location Management client (LMc) and a Location Management server (LMs), where the location information can be updated to or queried from the LMc. Optionally, there may be a Location Management proxy (LMp) between LMc and LMs.
With separation of control plane and data plane, the LM function is in the control plane. It may be a logical function at the control plane node, control plane anchor, or mobility controller. It may be distributed or centralized.

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

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

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

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

3. Distributed Mobility Anchoring

3.1. Configurations for Different Networks

We next describe some configurations with multiple distributed anchors. To cover the widest possible spectrum of scenarios, we consider architectures in which the control and data planes are separated, as described in [I-D.ietf-dmm-deployment-models].

3.1.1. Network-based DMM

Figure 1 shows a general scenario for network-based distributed mobility management.

The main characteristics of a network-based DMM solution are:

- There are multiple data plane anchors, each with a FM-DP function.
- The control plane may either be distributed (not shown in the figure) or centralized (as shown in the figure).
- The control plane and the data plane (Control Plane Anchor -- CPA -- and Data Plane Anchor -- DPA) may be co-located or not. If the
CPA is co-located with the distributed DPAs, then there are multiple co-located CPA-DPA instances (not shown in the figure).

- An IP prefix/address IP1 (anchored to the DPA with IP address IPa1) is assigned for use to a MN. The MN uses this IP1 address to communicate with CNs (not shown in the figure).
- The location management (LM) function may be co-located or split (as shown in the figure) into a separate server (LMs) and a client (LMc). In this case, the LMs may be centralized whereas the LMc may be distributed or centralized.

![Network Diagram]

Figure 1: Network-based DMM configuration

3.1.2. Client-based DMM

Figure 2 shows a general scenario for client-based distributed mobility management. In this configuration, the mobile node performs Control Plane Node (CPN) and Data Plane Node (DPN) mobility functions, namely the forwarding management and location management (client) roles.
4. IP Mobility Handling in Distributed Anchoring Environments - Mobility Support Only When Needed

IP mobility support may be provided only when needed instead of being provided by default. Three cases can be considered:

- Nomadic case: no address continuity is required. The IP address used by the MN changes after a movement and traffic using the old address is disrupted. If session continuity is required, then it needs to be provided by a solution running at L4 or above.
- Mobility case, traffic redirection: address continuity is required. When the MN moves, the previous anchor still anchors the traffic using the old IP address, and forwards it to the new MN's location. The MN obtains a new IP address anchored to the new location, and preferably uses it for new communications, established while connected at the new location.
- Mobility case, anchor relocation: address continuity is required. In this case the route followed by the traffic is optimized, by using some means for traffic indirection to deviate from default routes.

A straightforward choice of mobility anchoring is the following: the MN's chooses as source IP address for packets belonging to an IP
flow, an address allocated by the network the MN is attached to when the flow was initiated. As such, traffic belonging to this flow traverses the MN’s mobility anchor [I-D.seite-dmm-dma] [I-D.ietf-dmm-pmipv6-dlif].

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

There may be multiple IP prefixes/addresses that an MN can select when initiating a flow. They may be from the same access network or different access networks. The network may advertise these prefixes with cost options [I-D.mccann-dmm-prefixcost] so that the mobile node may choose the one with the least cost. In addition, these IP prefixes/addresses may be of different types regarding whether mobility support is needed [I-D.ietf-dmm-ondemand-mobility]. A MN will need to choose which IP prefix/address to use for each flow according to whether it needs IP mobility support or not.

4.1. Nomadic case (no need of IP mobility): Changing to new IP prefix/address

When IP mobility support is not needed for a flow, the LM and FM functions are not utilized so that the configurations in Section 3.1 are simplified as shown in Figure 3.
When there is no need to provide IP mobility to a flow, the flow may use a new IP address acquired from a new network as the MN moves to the new network.

Regardless of whether IP mobility is needed, if the flow has not terminated before the MN moves to a new network, the flow may subsequently restart using the new IP address assigned from the new network.

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

In Figure 3, a flow initiated while the MN was using the IP prefix IP1 -- anchored to a previous access router AR1 in network Net1 -- has terminated before the MN moves to a new network Net2. After moving to Net2, the MN uses the new IP prefix IP2 -- anchored to a new access router AR2 in network Net2 -- to start a new flow. Packets may then be forwarded without requiring IP layer mobility support.

An example call flow is outlined in Figure 4. A MN attaches to AR1, which sends a router advertisement (RA) including information about the prefix assigned to MN, from which MN configures an IP address (IP1). This address is used for new communications, for example with a correspondent node (CN). If the MN moves to a new network and

---

Figure 3: Changing to a new IP address/prefix
attaches to AR2, the process is repeated (MN obtains a new IP address, IP2, from AR2). Since the IP address (IP1) configured at the previously visited network is not valid at the current attachment point, and any existing flows have to be reestablished using IP2.

![Diagram of flow restarting with new IP prefix/address](image)

4.2. Mobility case, traffic redirection

When IP mobility is needed for a flow, the LM and FM functions in Section 3.1 are utilized. There are two possible cases: (i) the initial anchor remains the anchor and forwards traffic to a new locator in the new network, and (ii) the mobility anchor (data plane function) is changed but binds the MN’s transferred IP address/prefix. The latter enables optimized routes but requires some data plane node that enforces rules for traffic indirection. Next, we focus on the first case. The second one is addressed in Section 4.3.

Mobility support can be provided by using mobility management methods, such as the several approaches surveyed in the academic papers ([Paper-Distributed.Mobility], [Paper-Distributed.Mobility.PMIP] and [Paper-Distributed.Mobility.Review]). After moving, a certain MN’s
traffic flow may continue using the IP prefix from the prior network of attachment. Yet, some time later, the application generating this traffic flow may be closed. If the application is started again, the new flow may not need to use the prior network’s IP address to avoid having to invoke IP mobility support. This may be the case where a dynamic IP prefix/address, rather than a permanent one, is used. Packets belonging to this flow may then use the new IP prefix (the one allocated in the network where the flow is being initiated). Routing is again kept simpler without employing IP mobility and will remain so as long as the MN which is now in the new network does not move again to another network.

Figure 5: A flow continues to use the IP prefix from its home network after MN has moved to a new network

An example call flow in this case is outlined in Figure 5. In this example, the AR1 plays the role of FM-DP entity and redirects the

<table>
<thead>
<tr>
<th>MN</th>
<th>AR1</th>
<th>AR2</th>
<th>CN</th>
</tr>
</thead>
</table>
| MN attaches to AR1: | acquires MN-ID and profile | --RS--
| | | <-----RA(IP1)--- |
| Assigned prefix IP1 | IP1 address configuration | <-Flow(IP1,IPcn,...)---------------------------> |
| | | MN detaches from AR1 |
| | | MN attaches to AR2 |
| | | --RS----------------- |
| | | (some IP mobility support solution) |
| | | <-Flow(IP1,IPcn,...)---------------------------> |
| | | +<-------------RA(IP2,IP1)-----------+<-------------RA(IP2,IP1)-----------+ |
| | | <-Flow(IP1,IPcn,...)---------------------------> |
| | | <-Flow(IP1,IPcn,...)---------------------------> |
| Assigned prefix IP2 | IP2 address configuration |
| | | Flow(IP1,IPcn) terminates |
| | | <-new Flow(IP2,IPcn,...)---------------------------> |

Figure 5: A flow continues to use the IP prefix from its home network after MN has moved to a new network

An example call flow in this case is outlined in Figure 5. In this example, the AR1 plays the role of FM-DP entity and redirects the
traffic (e.g., using an IP tunnel) to AR2. Another solution could be to place an FM-DP entity closer to the CN network to perform traffic steering to deviate from default routes (which will bring the packet to AR1 per default routing). The LM and FM functions are implemented as shown in Figure 6.

Multiple instances of DPAs (at access routers), which are providing IP prefixes to the MNs, are needed to provide distributed mobility anchoring in an appropriate configuration such as those described in Figure 1 (Section 3.1.1) for network-based distributed mobility or in Figure 2 (Section 3.1.2) for client-based distributed mobility.

4.3. Mobility case, anchor relocation

We focus next on the case where the mobility anchor (data plane function) is changed but binds the MN’s transferred IP address/prefix. This enables optimized routes but requires some data plane node that enforces rules for traffic indirection.

IP mobility is invoked to enable IP session continuity for an ongoing flow as the MN moves to a new network. Here the anchoring of the IP address of the flow is in the home network of the flow (i.e., different from the current network of attachment). A centralized mobility management mechanism may employ indirection from the anchor in the home network to the current network of attachment. Yet it may be difficult to avoid using an unnecessarily long route (when the route between the MN and the CN via the anchor in the home network is significantly longer than the direct route between them). An
alternative is to move the IP prefix/address anchoring to the new network.

The IP prefix/address anchoring may move without changing the IP prefix/address of the flow. Here the LM and FM functions in Figure 1 in Section 3.1 are implemented as shown in Figure 7.

As an MN with an ongoing session moves to a new network, the flow may preserve IP session continuity by moving the anchoring of the original IP prefix/address of the flow to the new network.

One way to accomplish such a move is to use a centralized routing protocol, but such a solution may present some scalability concerns and its applicability is typically limited to small networks. One example of this type of solution is described in [I-D.ietf-rtgwg-atn-bgp]. When a mobile associates with an anchor the anchor injects the mobile’s prefix into the global routing system. If the mobile moves to a new anchor, the old anchor withdraws the /64 and the new anchor injects it instead.

5. Security Considerations

Security protocols and mechanisms are employed to secure the network and to make continuous security improvements, and a DMM solution is required to support them [RFC7333].
In a DMM deployment [I-D.ietf-dmm-deployment-models] various attacks such as impersonation, denial of service, man-in-the-middle attacks need to be prevented.

6. IANA Considerations

This document presents no IANA considerations.

7. Contributors

Alexandre Petrescu and Fred Templin had contributed to earlier versions of this document regarding distributed anchoring for hierarchical network and for network mobility, although these extensions were removed to keep the document within reasonable length.

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

Valuable comments have been received from John Kaippallimalil, ChunShan Xiong, Dapeng Liu and Fred Templin. Dirk von Hugo, Byju Pularikkal, Pierrick Seite have generously provided careful review with helpful corrections and suggestions. Marco Liebsch and Lyle Bertz also performed very detailed and helpful reviews of this document.

8. References

8.1. Normative References


8.2. Informative References

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

[I-D.ietf-dmm-fpc-cpdp]

[I-D.ietf-dmm-ondemand-mobility]

[I-D.ietf-dmm-pmipv6-dlif]

[I-D.ietf-rtgw-atn-bgp]

[I-D.matsushima-stateless-uplane-vepc]
[I-D.mccann-dmm-prefixcost]

[I-D.sarikaya-dmm-for-wifi]

[I-D.seite-dmm-dma]

[I-D.yhkim-dmm-enhanced-anchoring]

[Paper-Distributed.Mobility]

[Paper-Distributed.Mobility.PMIP]

[Paper-Distributed.Mobility.Review]

[RFC6459]

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

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on December 20, 2018.
Copyright Notice

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

Table of Contents

1. Introduction ......................................................... 3
2. Terminology ......................................................... 4
3. FPC Design Objectives and Deployment ............................. 7
4. FPC Mobility Information Model .................................... 9
   4.1. Model Notation and Conventions ............................... 9
   4.2. Templates and Attributes ..................................... 12
   4.3. Attribute-Expressions ....................................... 13
   4.4. Attribute Value Types ....................................... 14
   4.5. Namespace and Format ....................................... 14
   4.6. Configuring Attribute Values ................................ 15
   4.7. Entity Configuration Blocks ................................ 16
   4.8. Information Model Checkpoint ................................. 17
   4.9. Information Model Components ................................ 18
      4.9.1. Topology Information Model ............................... 18
      4.9.2. Service-Group ........................................... 18
      4.9.3. Domain Information Model ................................. 20
      4.9.4. DPN Information Model .................................. 20
      4.9.5. Policy Information Model ................................ 21
      4.9.6. Mobility-Context Information Model ...................... 24
      4.9.7. Monitor Information Model ............................... 26
5. Protocol ............................................................... 27
   5.1. Protocol Messages and Semantics .............................. 27
      5.1.1. Configure Message ....................................... 30
      5.1.2. Monitor Messages ........................................ 36
   5.2. Protocol Operation ............................................. 38
      5.2.1. DPN Selection ............................................ 38
      5.2.2. Policy Creation and Installation ......................... 41
      5.2.3. Simple RPC Operation ..................................... 43
      5.2.4. Policy and Mobility on the Agent ......................... 51
      5.2.5. Monitor Example .......................................... 53
6. Templates and Command Sets ........................................ 55
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.
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain:</td>
<td>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.</td>
</tr>
<tr>
<td>DPN:</td>
<td>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.</td>
</tr>
<tr>
<td>FPC Client:</td>
<td>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.</td>
</tr>
<tr>
<td>Mobility Context:</td>
<td>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 predefined 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.</td>
</tr>
<tr>
<td>Mobility Session:</td>
<td>Traffic to/from a mobile node that is expected to survive reconnection events.</td>
</tr>
<tr>
<td>Monitor:</td>
<td>A reporting mechanism for a list of events that trigger notification messages from a FPC Agent to a FPC Client.</td>
</tr>
<tr>
<td>Policy:</td>
<td>A Policy determines the mechanisms for managing specific traffic flows or packets. Policies specify QoS, rewriting rules for</td>
</tr>
</tbody>
</table>

Matsushima, et al. Expires December 20, 2018

[Page 5]
packet processing, etc. A Policy consists of one or more rules. Each rule is composed of a Descriptor and Actions. The Descriptor in a rule identifies packets (e.g., traffic flows), and the Actions apply treatments to packets that match the Descriptor in the rule. Policies can apply to Domains, DPNs, Mobile Nodes, Service-Groups, or particular Flows on a Mobile Node.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4. FPC Mobility Information Model

The FPC information model includes the following components:

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

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

Figure 2: FPC Information Model structure

4.1. Model Notation and Conventions

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

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

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

**Figure 3: Model Notation - An Example**

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

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

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

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

```
+--[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.
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.
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-State 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:
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.
Each Service-Group element contains the following information:

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

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

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

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

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

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

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

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

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

4.9.3. Domain Information Model

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

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

Figure 10: Domain Information Model

Each Domain entry contains the following information:

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

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

4.9.4. DPN Information Model

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

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

Figure 11: DPN Information Model

Each DPN entry contains the following information:

DPN-Key: A unique Identifier of the DPN.

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

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

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

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

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

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

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

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

4.9.5. Policy Information Model

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

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

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

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

Figure 12: Policy Information Model

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.9.6. Mobility-Context Information Model

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.9.7. Monitor Information Model

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

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

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

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

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

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

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

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

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

5. Protocol

5.1. Protocol Messages and Semantics

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

Table 1: Client to Agent Messages

Each message contains a header with the following information:

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

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

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

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

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

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

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

Error-Tag: An error tag.

Error-App-Tag: Application specific error tag.

Error-Message: A message describing the error.

Error-Info: Any data required for the response.

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

Figure 15: Error Information Model

Two Agent to Client notifications are supported.

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

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

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

5.1.1.1. Edit Operation Types

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

Table 3: Configure Edit Operations

5.1.1.2. Edit Operation

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

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

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

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

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

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

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

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

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

Figure 16: Configure Request

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

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

6. Mobility Contexts

5.1.1.3. Reference Scope

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

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

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

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

5.1.1.4.1. Immediate Response

The Response MUST include the following:

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

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

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

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

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

Figure 17: Example Hierarchical Cache
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.
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
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",
    "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" ]
    } ]
  } ] }
}, ...
]
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.

```json
"configure": {
  "client-id": 0,
  "operation-id": 0,
  "edit": [
    {
      "edit-id": 0,
      "edit-type": "create",
      "target": "/policy-information-model/descriptor-template",
      "value": {
        "descriptor-template-key": "desc1",
        "descriptor-type": "all-traffic"
      }
    },
    {
      "edit-id": 1,
      "edit-type": "create",
      "target": "/policy-information-model/action-template",
      "value": {
        "action-template-key": "action1",
        "action-type": "drop"
      }
    },
    {
      "edit-id": 2,
      "edit-type": "create",
      "target": "/policy-information-model/rule-template",
      "value": {
        "rule-template-key": "deny-all",
        "descriptor-match-type": "and"
      }
    }
  ]
}
```
"descriptor-configuration" : [{
  "descriptor-template-key" : "all" }],
"action-configuration" : [{
  "action-template-key" : "deny",
  "action-order" : 0 }
}, {
  "edit-id" : 3,
  "edit-type" : "create",
  "target" : "/policy-information-model/
/policy-template",
  "value" : {
    "policy-template-key" : "policy1",
    "entity-state" : "configured",
    "rule-template" : [{
      "rule-template-key" : "deny-all",
      "precedence" : 0 }]
  }}],
} |<---(2)- Response --------|
{ |
  "agent-id" : "agent1","|
  "operation-id" : 0, |
  "result-status" : "ok" |
} |

Figure 22: Example Policy Installation (focus on FPC reference point)

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

Installation of the policy on dpn1 is shown in the following Figure.
The Policy-Status is set to "Active" to make it immediately usable.
Leaving the status as Configured would permit its installation on the
DPN without an ability to use it in a Mobility Context. Such a use
case is often referred to as policy pre-configuration.
This message uses an edit type of "create" to add the policy template directly to the installed DPN policy set.

5.2.3. Simple RPC Operation

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

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

All messages sent from a Client to an Agent MUST be acknowledged by the Agent. The response must include all edit status as well as subsequent edits, which indicates the result of processing the message, as part of the Configure response. In case the processing of the message results in a failure, the Agent sets the global
status, Error-Type and Error-Tag accordingly and MAY clear the
entity, e.g. Mobility-Context, which caused the failure, in the
response.

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

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

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

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

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

+-------Router--------+
[MAG-C1] [MAG-C2] [LMA-C] Client | FPC | Agent | DPN1
[Anchor]

---(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...}>] } ] }
        } ] } ] } ]
    }]
  }]
}

---(2)- Response --------->

"result-status": "ok",

---(2)- route add

agent-id": "agent1",
"operation-id": 3,
"result-status": "ok",
"esign" : {"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...> } ] } ] } ] } ] } ] } ]
] }]

"result-status": "ok",

---(2)- route add

agent-id": "agent1",
"operation-id": 3,
[MN handover]  --PBU---->  --(3) CONFIG(MODIFY)---->  
"configure" : {  --tun1 mod->  
  "client-id" : 0,  
  "operation-id" : 4,  
  "edit" : {  
    "edit-id" : 0,  
    "edit-type" : "merge",  
    "target" : "/mobility-context/ctxt1",  
    "value" : {  
      "mobility-context-key" : "ctxt1",  
      "dpn" : "[ {  
        "dpn-key" : "DPN1",  
        "service-data-flow" : [ {  
          "identifier" : 0,  
          "service-data-flow-policy-configuration" : [  
            {"policy-template-key" :  
              "dl-tunnel-with-qos",  
              "policy-configuration" : [  
                {"index" : 1,  
                  "tunnel" : <NEW tunnel info...>}]}
          ]}
        ]
      ]
    }
  }
}<---PBA------  --tun1 mod->  
<---(4) OK -----------  
{  
  "agent-id" : "agent1",  
  "operation-id" : 4,  
  "result-status" : "ok",
}
After reception of the Proxy Binding Update (PBU) at the LMA Control-Plane function (LMA-C), the LMA-C selects a suitable DPN, which serves as Data-Plane anchor to the mobile node’s (MN) traffic. The LMA-C adds a new logical Mobility-Context to the DPN to treat the MN’s traffic (1) and includes a Mobility-Context-Key (ctxt1) in the Configure command. The LMA-C identifies the selected Anchor DPN by including the associated DPN identifier.

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

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

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

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

```
+-------Router--------+ +-------+ +---------+
| MGN-C1 | MGN-C2 | LMA-C | FPC    | Anchor  |
|---------+---------+-------+--------+--------+
[MN detach] | PBU-----|--------|--------|--------|

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>configure</td>
<td>:</td>
<td></td>
<td>{</td>
<td></td>
</tr>
<tr>
<td>&quot;client-identifier&quot; : 0,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
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 --------|                      -route add>
|  "agent-id" : "agent1", "
|  "operation-id" : 0,
|  "result-status" : "ok",
|  "notify-follows" : "true",
|
------ PBA ------

(3) Configure-Result- Notification
|                      -route add>
|  "agent-id" : "agent1", "

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.

```
[ ifc1 goes down which is reported as event type 3 ]
```

```
[ ifc1 goes down which is reported as event type 3 ]
```

```
[ ifc1 goes down which is reported as event type 3 ]
```

```
[ ifc1 goes down which is reported as event type 3 ]
```
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 Desciption 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] 
  ++[(ip-address) Use-Assigned-Address] 
+-[To-Spec] <List> (0) 
  ++[(ip-address) IP-Address] <List> 
  ++[IP-Address-Range] <List> 
    +-[(ip-address) IP-Address-Start] 
    ++[(ip-address) IP-Address-End] 
  ++[IP-Address-Mask] <List> 
    +-[(ip-address) IP-Address:] 
    ++[(Unsigned 32) IP-Bit-Mask-Width:] 
  ++[(mac-address) MAC-Address] <List> 
    ++[MAC-Address-Mask] <List> 
      +-[(mac-address) MAC-Address:] 
      ++[(mac-address) 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] <List> 
  +-[(ICMP-Type] <List> 
  +-[(ETH-Option] <List> 
  +-[(ecn-ip-codepoint] <List> 
  +-[(flowlabel] flow-label] <List> 
  +-[(flow-label-range] <List>
6.3. Tunnel Templates

The Network Service Header is specified in [RFC8300].

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

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

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

For GRE, the gre-key MUST be present.

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

- local tunnel endpoint identifier (teid) - MUST be present if local-address is nonempty
- remote tunnel endpoint identifier (teid) - MUST be present if remote-address is nonempty

sequence-numbers-on - Indicates that sequence numbers will be used

Tunnels can be used as Next Hop and Descriptor values.
6.4. Action Templates

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

Drop action has no values.

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

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

PMIP QoS is specified in [RFC7222].

Figure 33: Action Templates

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.
Figure 35: 3GPP Mobility Templates

```yaml
-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 36: 3GPP Packet Filter Template (Descriptor)

The following Command Set values are supported for 3GPP.

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

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

- Configure sends Mobility-Context information which can include information of a sensitive or vulnerable nature in some network environments as described above.
- Monitor related RPC operations do not specifically provide sensitive or vulnerable information but care must be taken by users to avoid identifier values that expose sensitive or vulnerable information.
- Notifications MUST be treated with same level of protection and scrutiny as the operations they correspond to. For example, a Configure-Result-Notification provides the same information that is sent as part of the input and output of the Configure RPC operation.

General usage of FPC MUST consider the following:

- FPC Naming Section 4.5 permits arbitrary string values but a user MUST avoid placing sensitive or vulnerable information in those values.
- Policies that are very narrow and permit the identification of specific traffic, e.g. that of a single user, SHOULD be avoided.

9. IANA Considerations

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

Registrant Contact: The DMM WG of the IETF.
XML: N/A, the requested URI is an XML namespace.
This document registers the following YANG modules in the "YANG Module Names" registry [RFC6020].

```plaintext
name: ietf-dmm-fpc
prefix: fpc
reference: TBD1

name: 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]

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


11.2. Informative References

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

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


Appendix A. YANG Data Model for the FPC protocol

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

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

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

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

Mapping of templates to YANG are performed as follows:

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

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

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

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

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

Five modules are defined:

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

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

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

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

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

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

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

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

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

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

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

A.1. FPC YANG Model

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

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

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

import ietf-inet-types { prefix inet;  
revision-date 2013-07-15; }
import ietf-dmm-fpc-settingsex { prefix fpcbase;  
revision-date 2018-05-17; }
import ietf-diam-trafficclassifier { prefix rfc5777;  
revision-date 2018-05-17; }
import ietf-restconf { prefix rc;  
revision-date 2017-01-26; }
import ietf-yang-patch { prefix ypatch;  
revision-date 2017-02-22; }

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

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

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

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

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

revision 2018-05-17 {
//General Structures
grouping templatedef {
  leaf extensible {
    type boolean;
    description "Indicates if the template is extensible";
  }
  leaf-list static-attributes {
    type string;
    description "Attribute (Name) whose value cannot change";
  }
  leaf-list mandatory-attributes {
    type string;
    description "Attribute (Name) of optional attributes that MUST be present in instances of this template.";
  }
  leaf entity-state {
    type enumeration {
      enum initial {
        description "Initial Configuration";
      }
      enum partially-configured {
        description "Partial Configuration";
      }
      enum configured {
        description "Configured";
      }
      enum active {
        description "Active";
      }
    }
    default initial;
    description "Entity State";
  }
  leaf version {
    type uint32;
    description "Template Version";
  }
  description "Template Definition";
}
typedef fpc-identity {
  type union {
    type uint32;
    type instance-identifier;
  }
}
type string;
}
description "FPC Identity";
}
grouping index {
  leaf index {
    type uint16;
    description "Index";
  }
  description "Index Value";
}

// Policy Structures
grouping descriptor-template-key {
  leaf descriptor-template-key {
    type fpc:fpc-identity;
    mandatory true;
    description "Descriptor Key";
  }
  description "Descriptor-Template Key";
}
grouping action-template-key {
  leaf action-template-key {
    type fpc:fpc-identity;
    mandatory true;
    description "Action Key";
  }
  description "Action-Template Key";
}
grouping rule-template-key {
  leaf rule-template-key {
    type fpc:fpc-identity;
    mandatory true;
    description "Rule Identifier";
  }
  description "Rule Key";
}
grouping policy-template-key {
  leaf policy-template-key {
    type fpc:fpc-identity;
    mandatory true;
    description "Rule Identifier";
  }
  description "Rule Key";
}

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

// 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";
    }
    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 "Monitor Configuration";

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

canterior 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:asename-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-notification augment";
}

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

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

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

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

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

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

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

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

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

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

Matsushima, et al. Expires December 20, 2018
description
"This module contains YANG definition for
Forwarding Policy Configuration Protocol (FPCP).
It contains Settings definitions as well as Descriptor and
Action extensions.

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

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

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

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

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

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

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

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

enum dual {
    value 2;
    description "IPv4 and IPv6"
}

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-directional";
}

description "Packet Filter Direction";

typedef component-type-id {
    type uint8 {
        range "16 | 17 | 32 | 33 | 35 | 48 | 64 | 65 |
        + " 80 | 81 | 96 | 112 | 128";
    } 
    description "Specifies the Component Type";
}

grouping packet-filter {
    leaf direction {
        type 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 Port Min Value";
    }
    leaf remote-port-hi {
        type inet:port-number;
        description "Remote Port Max Value";
    }
}

leaf ipsec-index {
    type traffic-selectors:ipsec-spi;
}
description "IPSec Index";
}
leaf traffic-class {
  type inet:dscp;
  description "Traffic Class";
}
case traffic-class-range {
  leaf traffic-class-lo {
    type inet:dscp;
    description "Traffic Class Min Value";
  }
  leaf traffic-class-hi {
    type inet:dscp;
    description "Traffic Class Max Value";
  }
}
leaf-list flow-label {
  type inet:ipv6-flow-label;
  description "Flow Label";
}
description "Component Value";
}
}
description "Packet Filter";
}
grouping prefix-descriptor {
  leaf destination-ip {
    type inet:ip-prefix;
    description "Rule of destination IP";
  }
  leaf source-ip {
    type inet:ip-prefix;
    description "Rule of source IP";
  }
  description "Traffic descriptor based upon source/
    destination as IP prefixes";
}
grouping fpc-descriptor-value {
  choice descriptor-value {
    mandatory true;
    leaf all-traffic {
      type empty;
      description "admit any";
    }
    leaf no-traffic {
      type empty;
    }
  }
}

description "deny any";
}
case prefix-descriptor {
    uses fpcbase:prefix-descriptor;
    description "IP Prefix descriptor";
}
case pmip-selector {
    uses traffic-selectors:traffic-selector;
    description "PMIP Selector";
}
container rfc5777-classifier-template {
    uses rfc5777:classifier;
    description "RFC 5777 Classifier";
}
container packet-filter {
    uses fpcbase:packet-filter;
    description "Packet Filter";
}
case tunnel-info {
    uses fpcbase:tunnel-value;
    description "Tunnel Descriptor (only considers source info)";
}
description "Descriptor Value";
}
description "FPC Descriptor Values";

// Next Hop Structures
typedef fpc-service-path-id {
    type uint32 {
        range "0..33554431";
    }
    description "SERVICE_PATH_ID";
}
typedef fpc-mpls-label {
    type uint32 {
        range "0..1048575";
    }
    description "MPLS label";
}
typedef segment-id {
    type string {
        length "16";
    }
    description "SR Segment Identifier";
}
grouping fpc-nexthop {

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

////////////////////////////////////////////////////////////////////////
// 3GPP Integration         //

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

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

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

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

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

This module references [RFC6991].

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


    prefix "qos-pmip";

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

    import ietf-trafficselector-types { prefix traffic-selectors; 

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

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

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

typedef sr-id {
    type uint8;
}

// QoS Option Field Type Definitions
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";
}
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
      "Reserved values: (6) to (255) Currently not used. Receiver MUST ignore the option
      received with any value in this range."
  }
}
description
"This flag is used to request that the uplink/downlink
flows for which the network is providing
Guaranteed-Bit-Rate service be excluded from the
target IP flows for which
Per-Session-Agg-Max-UL/DL-Bit-Rate is measured.";
}
description "Per-Session-Agg-Max-Bit-Rate Value";
}
grouping Allocation-Retention-Priority-Value {
leaf priority-level {
  type uint8 {
    range "0..15";
  }
  mandatory true;
description
"This is a 4-bit unsigned integer value. It is used to decide
whether a mobility session establishment or modification
request can be accepted; this is typically used for
admission control of Guaranteed Bit Rate traffic in case of
resource limitations.";
}
leaf preemption-capability {
  type enumeration {
    enum enabled {
      value 0;
      description "enabled";
    }
    enum disabled {
      value 1;
      description "disabled";
    }
    enum reserved1 {
      value 2;
      description "reserved1";
    }
    enum reserved2 {
      value 3;
      description "reserved2";
    }
  }
  mandatory true;
description
"This is a 2-bit unsigned integer value. It defines whether a
service data flow can get resources that were already
assigned to another service data flow with a lower priority
level.";
leaf preemption-vulnerability {
    type enumeration {
        enum enabled {
            value 0;
            description "enabled";
        } enum disabled {
            value 1;
            description "disabled";
        } enum reserved1 {
            value 2;
            description "reserved1";
        } enum reserved2 {
            value 3;
            description "reserved2";
        }
    } mandatory true;
    description "This is a 2-bit unsigned integer value. It defines whether a service data flow can lose the resources assigned to it in order to admit a service data flow with a higher priority level.";
    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;
}
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";
  }
  description
    "The guaranteed bandwidth in bits per second for downlink
    IP flows. The measurement units are bits per second.";
}
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;
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>";

container traffic-selector {
  uses traffic-selectors:traffic-selector;
  description "traffic selector";
}

description "PMIP QoS Option";
description
"This module contains a collection of YANG definitions for traffic selectors for flow bindings.

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

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

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

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

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

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

// Type definitions and groupings
typedef ipsec-spi {
  type uint32;
  description
    "The first 32-bit IPsec Security Parameter Index (SPI) value on data. This field is defined in [RFC4303].";
}
grouping traffic-selector-base {
  description "A grouping of the common leaves between the
  v4 and v6 Traffic Selectors";
  container ipsec-spi-range {
    presence "Enables setting ipsec spi range";
    description "Inclusive range representing IPSec Security Parameter
    Indices to be used. When only start-spi is present, it
    represents a single spi.";
    leaf start-spi {
      type ipsec-spi;
      mandatory true;
      description "The first 32-bit IPsec SPI value on data.";
    }
    leaf end-spi {
      type ipsec-spi;
      must ". >= ../start-spi" {
        error-message
        "The end-spi must be greater than or equal
to start-spi";
      }
      description "If more than one contiguous SPI value needs to be matched,
then this field indicates the end value of a range.";
    }
  }
  container source-port-range {
    presence "Enables setting source port range";
    description "Inclusive range representing source ports to be used.
When only start-port is present, it represents a single
port. These value(s) are from the range of port numbers
defined by IANA (http://www.iana.org).";
    leaf start-port {
      type inet:port-number;
      mandatory true;
      description "The first 16-bit source port number to be matched";
    }
    leaf end-port {
      type inet:port-number;
      must ". >= ../start-port" { 

error-message
  "The end-port must be greater than or equal to start-port";
} 
description
  "The last 16-bit source port number to be matched";
}
}
container destination-port-range {
  presence "Enables setting destination port range";
  description
  "Inclusive range representing destination ports to be used. When only start-port is present, it represents a single port.";
  leaf start-port {
    type inet:port-number;
    mandatory true;
    description
    "The first 16-bit destination port number to be matched";
  }
  leaf end-port {
    type inet:port-number;
    must ". >= ../start-port" {
      error-message
      "The end-port must be greater than or equal to start-port";
    }
    description
    "The last 16-bit destination port number to be matched";
  }
}
}
grouping ipv4-binary-traffic-selector {
  container source-address-range-v4 {
    presence "Enables setting source IPv4 address range";
    description
    "Inclusive range representing IPv4 addresses to be used. When only start-address is present, it represents a single address.";
    leaf start-address {
      type inet:ipv4-address;
      mandatory true;
      description
      "The first source address to be matched";
    }
    leaf end-address {
      type inet:ipv4-address;
      description
      "The last source address to be matched";
    }
  }
}

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

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

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

description "ipv4 binary traffic selector";
}

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

type inet:ipv6-address;
mandatory true;
description
"The first destination address, from the
range of 128-bit IPv6 addresses to be matched";
}
leaf end-address {
    type inet:ipv6-address;
description
    "The last destination address, from the
    range of 128-bit IPv6 addresses to be matched";
}
}
container flow-label-range {
    presence "Enables setting Flow Label range";
description
    "Inclusive range representing IPv4 addresses to be used. When
    only start-flow-label is present, it represents a single
    flow label.";
    leaf start-flow-label {
        type inet:ipv6-flow-label;
description
        "The first flow label value to be matched";
    }
    leaf end-flow-label {
        type inet:ipv6-flow-label;
must ". >= ../start-flow-label" {
            error-message
            "The end-flow-label 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
leaf end-traffic-class {
  type inet:dscp;
  must ". >= ../start-traffic-class" {
    error-message
    "The end-traffic-class must be greater than or equal to start-traffic-class";
  }
  description
  "The last traffic class value to be matched";
}

container next-header-range {
  presence "Enables setting Next Header range";
  description
  "Inclusive range representing Next Headers to be used. When only start-next-header is present, it represents a single Next Header."
  leaf start-next-header {
    type uint8;
    description
    "The first 8-bit next header value to be matched."
  }
  leaf end-next-header {
    type uint8;
    must ". >= ../start-next-header" {
      error-message
      "The end-next-header must be greater than or equal to start-next-header";
    }
    description
    "The last 8-bit next header value to be matched.";
  }
  description "ipv6 binary traffic selector";
}

grouping traffic-selector {
  leaf ts-format {
    type identityref {
      base traffic-selector-format;
    }
    description "Traffic Selector Format";
  }
  uses traffic-selectors:traffic-selector-base;
  uses traffic-selectors:ipv4-binary-traffic-selector;
  uses traffic-selectors:ipv6-binary-traffic-selector;
}
description
"The traffic selector includes the parameters used to match packets for a specific flow binding."

reference
"RFC 6089: Flow Bindings in Mobile IPv6 and Network Mobility (NEMO) Basic Support"

A.5. RFC 5777 Classifier YANG Model

This module defines the RFC 5777 Classifier.

This module references [RFC5777].

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

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

  prefix "diamclassifier";

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

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

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

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

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

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

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

  Editor: Lyle Bertz

  Matsushima, et al.   Expires December 20, 2018
<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";
}
}
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 "Fragmentation 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";
  }
  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* [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* [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
Internet-Draft              DMM FPC Protocol                   June 2018

---:(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 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 ibi?  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)
---: (nsh)
  ---: (interface)
  ---: (segment-identifier)
    ---: (mpls-label-stack)
      ---: (mpls-sr-stack)
        ---: (srv6-stack)
      ---: (mpls-label-stack*)
    ---: (mpls-sr-stack*)
---: (tunnel-info)
  ---: tunnel-local-address?
  ---: tunnel-remote-address?
  ---: mtu-size?
  ---: tunnel?
  ---: payload-type?
  ---: gre-key?
  ---: gtp-tunnel-info
    ---: local-tunnel-identifier?
    ---: remote-tunnel-identifier?
    ---: sequence-numbers-enabled?
  ---: ebi?
  ---: lbi?
+++: (qos)
  ---: trafficclass?
  ---: per-mn-agg-max-dl?
    qos-pmip:Per-MN-Agg-Max-DL-Bit-Rate-Value
  ---: per-mn-agg-max-ul?
    qos-pmip:Per-MN-Agg-Max-UL-Bit-Rate-Value
  ---: per-session-agg-max-dl
    ---: max-rate
text
  ---: per-session-agg-max-ul
    ---: max-rate
text
  ---: priority-level
text
+++: (conclusion)
Internet-Draft              DMM FPC Protocol                   June 2018

Internet-Draft  DMM FPC Protocol  June 2018

|  +--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
Internet-Draft              DMM FPC Protocol                   June 2018

+---(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
+++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:
  +++ 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
    +++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
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

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

Service-Group resides under the Topology-Information-Mode

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

Scrubbed YANG for 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
Abstract

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

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

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on August 2, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the
document authors. All rights reserved.

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

Table of Contents

1. Introduction .................................................. 3
2. Terminology .................................................. 4
3. PMIPv6 DMM extensions ........................................ 5
   3.1. Initial registration ..................................... 7
   3.2. The CMD as PBU/PBA relay .............................. 8
   3.3. The CMD as MAAR locator .............................. 11
   3.4. The CMD as MAAR proxy ................................ 12
   3.5. De-registration ........................................ 13
   3.6. The Distributed Logical Interface (DLIF) concept .......... 13
4. Message Format .................................................. 17
   4.1. Proxy Binding Update .................................. 17
   4.2. Proxy Binding Acknowledgment .......................... 18
   4.3. Anchored Prefix Option ............................... 19
   4.4. Local Prefix Option ................................... 20
   4.5. Previous MAAR Option .................................. 21
   4.6. Serving MAAR Option ................................... 22
   4.7. DLIF Link-local Address Option ....................... 23
   4.8. DLIF Link-layer Address Option ....................... 24
5. IANA Considerations ............................................. 25
6. Security Considerations ......................................... 25
7. Acknowledgments ................................................ 25
1. Introduction

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

Current IP mobility solutions, standardized with the names of Mobile IPv6 [RFC6275], or Proxy Mobile IPv6 (PMIPv6) [RFC5213], just to cite the two most relevant examples, offer mobility support at the cost of handling operations at a cardinal point, the mobility anchor (i.e., the home agent for Mobile IPv6, and the local mobility anchor for Proxy Mobile IPv6), and burdening it with data forwarding and control mechanisms for a great amount of users. As stated in [RFC7333], centralized mobility solutions are prone to several problems and limitations: longer (sub-optimal) routing paths, scalability problems, signaling overhead (and most likely a longer associated handover latency), more complex network deployment, higher vulnerability due to the existence of a potential single point of failure, and lack of granularity of the mobility management service (i.e., mobility is offered on a per-node basis, not being possible to define finer granularity policies, as for example per-application).

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

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

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

2. Terminology

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

- Local Mobility Anchor (LMA)
- Mobile Access Gateway (MAG)
- Mobile Node (MN)
- Binding Cache Entry (BCE)
- Proxy Care-of Address (P-CoA)
- Proxy Binding Update (PBU)
- Proxy Binding Acknowledgement (PBA)

The following terms used in this document are defined in the DMM Deployment Models and Architectural Considerations document [I-D.ietf-dmm-deployment-models]:

- Home Control-Plane Anchor (Home-CPA)
- Home Data Plane Anchor (Home-DPA)
- Access Control Plane Node (Access-CPN)
Access Data Plane Node (Access-DPN)

The following terms are defined and used in this document:

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

CMD (Central Mobility Database). The node that stores the BCEs allocated for the MNs in the mobility domain. It plays the role of Home-CPA.

P-MAAR (Previous MAAR). When a MN moves to a new point of attachment a new MAAR might be allocated as its anchor point for future IPv6 prefixes. The MAAR that served the MN prior to new attachment becomes the P-MAAR. It is still the anchor point for the IPv6 prefixes it had allocated to the MN in the past and serves as the Home-DPA for flows using these prefixes. There might be several P-MAARs serving a MN when the MN is frequently switching points of attachment while maintaining long-lasting flows.

S-MAAR (Serving MAAR). The MAAR which the MN is currently attached to. Depending on the prefix, it plays the role of Access-DPN, Home-DPA and Access-CPN.

DLIF (Distributed Logical Interface). It is a logical interface at the IP stack of the MAAR. For each active prefix used by the MN, the S-MAAR has a DLIF configured (associated to each MAAR still anchoring flows). In this way, an S-MAAR exposes itself towards each MN as multiple routers, one as itself and one per P-MAAR.

3. PMIPv6 DMM extensions

The solution consists of de-coupling the entities that participate in the data and the control planes: the data plane becomes distributed and managed by the MAARs near the edge of the network, while the control plane, besides those on the MAARs, relies on a central entity called Central Mobility Database (CMD). In the proposed architecture, the hierarchy present in PMIPv6 between LMA and MAG is preserved, but with the following substantial variations:

- The LMA is relieved from the data forwarding role, only the Binding Cache and its management operations are maintained. Hence the LMA is renamed into Central Mobility Database (CMD), which is therefore a Home-CPA. Also, the CMD is able to send and parse both PBU and PBA messages.
The MAG is enriched with the LMA functionalities, hence the name Mobility Anchor and Access Router (MAAR). It maintains a local Binding Cache for the MNs that are attached to it and it is able to send and parse PBU and PBA messages.

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

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

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

- The CMD is a PBU/PBA relay;
- The CMD is only a MAAR locator;
- The CMD is a PBU/PBA proxy.

This solution can be categorized under Model-1: Split Home Anchor Mode in [I-D.ietf-dmm-deployment-models]. As another note, the solution described in this document allows performing per-prefix anchoring decisions, to support e.g., some flows to be anchored at a central Home-DPA (like a traditional LMA) or to enable an application to switch to the locally anchored prefix to gain route optimization, as indicated in [I-D.ietf-dmm-ondemand-mobility]. This type of per-prefix treatment would potentially require additional extensions to the MAARs and signaling between the MAARs and the MNs to convey the per-flow anchor preference (central, distributed), which are not covered in this document.
Note that a MN MAY move across different MAARs, which might result in several P-MAARs existing at a given moment of time, each of them anchoring a different prefix used by the MN.

3.1. Initial registration

Initial registration is performed when an MN attaches to a network for the first time (rather than attaching to a new network after moving from a previous one).

In this description (shown in Figure 1), it is assumed that:

1. The MN is attaching to MAAR1.
2. The MN is authorized to attach to the network.

Upon MN attachment, the following operations take place:

1. MAAR1 assigns an IPv6 global prefix from its own prefix pool to the MN (Pref1). It also stores this prefix (Pref1) in the locally allocated temporary Binding Cache Entry (BCE).
2. MAAR1 sends a PBU [RFC5213] with Pref1 and the MN’s MN-ID to the CMD.
3. Since this is an initial registration, the CMD stores a permanent BCE containing as primary fields the MN-ID, Pref1 and MAAR1’s address as a Proxy-CoA.
4. The CMD replies with a PBA with the usual options defined in PMIPv6 [RFC5213], meaning that the MN’s registration is fresh and no past status is available.
5. MAAR1 stores the BCE described in (1) and unicasts a Router Advertisement (RA) to the MN with Pref1.
6. The MN uses Pref1 to configure an IPv6 address (IP1) (e.g., with stateless auto-configuration, SLAAC).

Note that:

1. Alternative IPv6 auto-configuration mechanisms can also be used, though this document describes the SLAAC-based one.
2. IP1 is routable at MAAR1, in the sense that it is on the path of packets addressed to the MN.
3. MAAR1 acts as a plain router for packets destined to the MN, as no encapsulation nor special handling takes place.

In the diagram shown in Figure 1 (and subsequent diagrams), the flow of packets is presented using ‘*’.

```
+-----+      +---+                +--+
|MAAR1|      |CMD|                |CN|
+-----+      +---+                +*-+

|MN attach. |                   *        |++++|
|           |                   flow1 *   |
|local BCE  |                   *          |
|allocation |                   *          |

--- PBU --> |                   *        |
|BCE creation|                   MAAR1+----MAAR2+-----MAAR3 |
|<-- PBA ---|                   *          |
|local BCE  |                   *          |
|finalized  |                   *          |

Operations sequence         Packets flow

Figure 1: First attachment to the network

Note that the registration process does not change regardless of the CMD’s modes (relay, locator or proxy) described next. The procedure is depicted in Figure 1.

3.2. The CMD as PBU/PBA relay

Upon MN mobility, if the CMD behaves as PBU/PBA relay, the following operations take place:

1. When the MN moves from its current point of attachment and attaches to MAAR2 (now the S-MAAR), MAAR2 reserves another IPv6 prefix (Pref2), it stores a temporary BCE, and it sends a plain PBU to the CMD for registration.

2. Upon PBU reception and BC lookup, the CMD retrieves an already existing entry for the MN, binding the MN-ID to its former location; thus, the CMD forwards the PBU to the MAAR indicated as
Proxy CoA (MAAR1), including a new mobility option to communicate the S-MAAR’s global address to MAAR1, defined as Serving MAAR Option in Section 4.6. The CMD updates the P-CoA field in the BCE related to the MN with the S-MAAR’s address.

3. Upon PBU reception, MAAR1 can install a tunnel on its side towards MAAR2 and the related routes for Pref1. Then MAAR1 replies to the CMD with a PBA (including the option mentioned before) to ensure that the new location has successfully changed, containing the prefix anchored at MAAR1 in the Home Network Prefix option.

4. The CMD, after receiving the PBA, updates the BCE populating an instance of the P-MAAR list. The P-MAAR list is an additional field on the BCE that contains an element for each P-MAAR involved in the MN’s mobility session. The list element contains the P-MAAR’s global address and the prefix it has delegated (see Appendix B for further details). Also, the CMD sends a PBA to the new S-MAAR, containing the previous Proxy-CoA and the prefix anchored to it embedded into a new mobility option called Previous MAAR Option (defined in Section 4.5), so that, upon PBA arrival, a bi-directional tunnel can be established between the two MAARs and new routes are set appropriately to recover the IP flow(s) carrying Pref1.

5. Now packets destined to Pref1 are first received by MAAR1, encapsulated into the tunnel and forwarded to MAAR2, which finally delivers them to their destination. In uplink, when the MN transmits packets using Pref1 as source address, they are sent to MAAR2, as it is MN’s new default gateway, then tunneled to MAAR1 which routes them towards the next hop to destination. Conversely, packets carrying Pref2 are routed by MAAR2 without any special packet handling both for uplink and downlink.
For MN’s next movements the process is repeated except the number of P-MAARs involved increases (accordingly to the number of prefixes that the MN wishes to maintain). Indeed, once the CMD receives the first PBU from the new S-MAAR, it forwards copies of the PBU to all the P-MAARs indicated in the BCE as current P-CoA (i.e., the MAAR prior to handover) and in the P-MAARs list. They reply with a PBA to the CMD, which aggregates them into a single one to notify the S-MAAR, that finally can establish the tunnels with the P-MAARs.

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

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

Figure 3: Scenario after a handover, CMD as locator
3.4. The CMD as MAAR proxy

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

![Diagram showing data packets flow and operations sequence for PMIPv6 DMM and DLIF](image-url)

**Figure 4: Scenario after a handover, CMD as proxy**
3.5. De-registration

The de-registration mechanism devised for PMIPv6 cannot be used as is in this solution. The reason for this is that each MAAR handles an independent mobility session (i.e., a single or a set of prefixes) for a given MN, whereas the aggregated session is stored at the CMD. Indeed, when a previous MAAR initiates a de-registration procedure, because the MN is no longer present on the MAAR’s access link, it removes the routing state for that (those) prefix(es), that would be deleted by the CMD as well, hence defeating any prefix continuity attempt. The simplest approach to overcome this limitation is to deny a P-MAAR to de-register a prefix, that is, allowing only a serving MAAR to de-register the whole MN session. This can be achieved by first removing any layer-2 detachment event, so that de-registration is triggered only when the session lifetime expires, hence providing a guard interval for the MN to connect to a new MAAR. Then, a change in the MAAR operations is required, and at this stage two possible solutions can be deployed:

- A previous MAAR stops the BCE timer upon receiving a PBU from the CMD containing a "Serving MAAR" option. In this way only the Serving MAAR is allowed to de-register the mobility session, arguing that the MN definitely left the domain.
- Previous MAARs can, upon BCE expiry, send de-registration messages to the CMD, which, instead of acknowledging the message with a 0 lifetime, sends back a PBA with a non-zero lifetime, hence renewing the session, if the MN is still connected to the domain.

3.6. The Distributed Logical Interface (DLIF) concept

One of the main challenges of a network-based DMM solution is how to allow a mobile node to simultaneously send/receive traffic which is anchored at different MAARs, and how to influence on the mobile node’s selection process of its source IPv6 address for a new flow, without requiring special support from the mobile node’s IP stack. This document defines the Distributed Logical Interface (DLIF), which is a software construct that allows to easily hide the change of associated anchors from the mobile node.
The basic idea of the DLIF concept is the following: each serving MAAR exposes itself towards a given MN as multiple routers, one per P-MAAR associated to the MN. Let’s consider the example shown in Figure 5, MN1 initially attaches to MAAR1, configuring an IPv6 address (prefA::MN1) from a prefix locally anchored at MAAR1 (prefA::/64). At this stage, MAAR1 plays both the role of anchoring and serving MAAR, and also behaves as a plain IPv6 access router. MAAR1 creates a distributed logical interface to communicate (point-to-point link) with MN1, exposing itself as a (logical) router with a specific MAC (e.g., 00:11:22:33:01:01) and IPv6 addresses (e.g., prefA::MAAR1/64 and fe80:211:22ff:fe33:101/64) using the DLIF mn1mar1. As explained below, these addresses represent the "logical" identity of MAAR1 towards MN1, and will "follow" the mobile node while roaming within the domain (note that the place where all this information is maintained and updated is out-of-scope of this draft; potential examples are to keep it on the home subscriber server -- HSS -- or the user’s profile).

If MN1 moves and attaches to a different MAAR of the domain (MAAR2 in the example of Figure 5), this MAAR will create a new logical interface (mn1mar2) to expose itself towards MN1, providing it with a locally anchored prefix (prefB::/64). In this case, since the MN1 has another active IPv6 address anchored at a MAAR1, MAAR2 also needs
to create an additional logical interface configured to exactly resemble the one used by MAAR1 to communicate with MN1. In this example, there is only one P-MAAR (in addition to MAAR2, which is the serving one): MAAR1, so only the logical interface mn1mar1 is created, but the same process would be repeated in case there were more P-MAARs involved. In order to maintain the prefix anchored at MAAR1 reachable, a tunnel between MAAR1 and MAAR2 is established and the routing is modified accordingly. The PBU/PBA signaling is used to set-up the bi-directional tunnel between MAAR1 and MAAR2, and it might also be used to convey to MAAR2 the information about the prefix(es) anchored at MAAR1 and about the addresses of the associated DLIF (i.e., mn1mar1).

Figure 6 shows the logical interface concept in more detail. The figure shows two MAARs and three MNs. MAAR1 is currently serving MN2 and MN3, while MAAR2 is serving MN1. MN1, MN2 and MN3 have two P-MAARs: MAAR1 and MAAR2. Note that a serving MAAR always plays the role of anchoring MAAR for the attached (served) MNs. Each MAAR has one single physical wireless interface.

As introduced before, each MN always "sees" multiple logical routers -- one per P-MAAR -- independently of its currently serving MAAR. From the point of view of the MN, these MAARs are portrayed as different routers, although the MN is physically attached to one single interface. The way this is achieved is by the serving MAAR
configuring different logical interfaces. Focusing on MN1, it is currently attached to MAAR2 (i.e., MAAR2 is its serving MAAR) and, therefore, it has configured an IPv6 address from MAAR2’s pool (e.g., prefB::/64). MAAR2 has set-up a logical interface (mn1mar2) on top of its wireless physical interface (phy if MAAR2) which is used to serve MN1. This interface has a logical MAC address (LMAC6), different from the hardware MAC address (HMAC2) of the physical interface of MAAR2. Over the mn1mar2 interface, MAAR2 advertises its locally anchored prefix prefB::/64. Before attaching to MAAR2, MN1 was attached to MAAR1, configuring also an address locally anchored at that MAAR, which is still being used by MN1 in active communications. MN1 keeps "seeing" an interface connecting to MAAR1, as if it were directly connected to the two MAARs. This is achieved by the serving MAAR (MAAR2) configuring an additional distributed logical interface: mn1mar1, which behaves exactly as the logical interface configured by MAAR1 when MN1 was attached to it. This means that both the MAC and IPv6 addresses configured on this logical interface remain the same regardless of the physical MAAR which is serving the MN. The information required by a serving MAAR to properly configure this logical interfaces can be obtained in different ways: as part of the information conveyed in the PBA, from an external database (e.g., the HSS) or by other means. As shown in the figure, each MAAR may have several logical interfaces associated to each attached MN, having always at least one (since a serving MAAR is also an anchoring MAAR for the attached MN).

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

The distributed logical interface concept also enables the following use case: suppose that access to a local IP network is provided by a given MAAR (e.g., MAAR1 in the example shown in Figure 5) and that the resources available at that network cannot be reached from outside the local network (e.g., cannot be accessed by an MN attached to MAAR2). This is similar to the local IP access scenario considered by 3GPP, where a local gateway node is selected for sessions requiring access to services provided locally (instead of going through a central gateway). The goal is to allow an MN to be able to roam while still being able to have connectivity to this local IP network. The solution adopted to support this case makes
use of RFC 4191 [RFC4191] more specific routes when the MN moves to a
MAAR different from the one providing access to the local IP network
(MAAR1 in the example). These routes are advertised through the
distributed logical interface representing the MAAR providing access
to the local network (MAAR1 in this example). In this way, if MN1
moves from MAAR1 to MAAR2, any active session that MN1 may have with
a node on the local network connected to MAAR1 will survive via the
tunnel between MAAR1 and MAAR2. Also, any potential future
connection attempt towards the local network will be supported, even
though MN1 is no longer attached to MAAR1.

4.  Message Format

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

4.1.  Proxy Binding Update

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

0               1               2               3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Sequence #                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|A|H|L|K|M|R|P|F|T|B|S|D Reser |            Lifetime           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                               |
| Mobility options                                             |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

MAAR Flag (D)

The D Flag is set to indicate to the receiver of the message that
the Proxy Binding Update is from a MAAR. When an LMA that does
not support the extensions described in this document receives a
message with the D-Flag set, the PBU in that case MUST NOT be
processed by the LMA and an error MUST be returned.

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

4.2. Proxy Binding Acknowledgment

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

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Status      |K|R|P|T|B|S|D| |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Sequence #            |           Lifetime            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                               |
| Mobility options                       |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

MAAR (D)

The D is set to indicate that the sender of the message supports operating as a Mobility Anchor and Access Router. When a MAG that does not support the extensions described in this document receives a message with the D-Flag set, it MUST ignore the message and an error MUST be returned.

Mobility Options

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

A new Anchored Prefix option is defined for use with the Proxy Binding Update and Proxy Binding Acknowledgment messages exchanged between MAARs and CMDs. Therefore, this option can only appear if the D bit is set in a PBU/PBA. This option is used for exchanging the mobile node’s prefix anchored at the anchoring MAAR. There can be multiple Anchored Prefix options present in the message.

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

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |   Length      |   Reserved    | Prefix Length |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                               |
|                        Anchored Prefix                        |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

**Type**

IAANA-1.

**Length**

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

**Reserved**

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

**Prefix Length**

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

**Anchored Prefix**
A sixteen-byte field containing the mobile node’s IPv6 Anchored Prefix. Only the first Prefix Length bytes are valid for the Anchored Prefix. The rest of the bytes MUST be ignored.

4.4. Local Prefix Option

A new Local Prefix option is defined for use with the Proxy Binding Update and Proxy Binding Acknowledgment messages exchanged between MAARs. Therefore, this option can only appear if the D bit is set in a PBU/PBA. This option is used for exchanging a prefix of a local network that is only reachable via the anchoring MAAR. There can be multiple Local Prefix options present in the message.

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

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

Type

IANA-2.

Length

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

Reserved

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

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

Local Prefix

A sixteen-byte field containing the IPv6 Local Prefix. Only the first Prefix Length bytes are valid for the IPv6 Local Prefix. The rest of the bytes MUST be ignored.

4.5. Previous MAAR Option

This new option is defined for use with the Proxy Binding Acknowledgement messages exchanged by the CMD to a MAAR. This option is used to notify the S-MAAR about the previous MAAR’s global address and the prefix anchored to it. There can be multiple Previous MAAR options present in the message. Its format is as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------------------------------+-
|      Type     |     Length    | Prefix Length |
+-------------------------------+-
|                        P-MAAR’s address |
+-------------------------------+-
|                        Home Network Prefix |
+-------------------------------+-
```

Type

IANA-3.

Length

8-bit unsigned integer indicating the length of the option in octets, excluding the type and length fields. This field MUST be set to 33.
Prefix Length

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

Previous MAAR’s address

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

Home Network Prefix

A sixteen-byte field containing the mobile node’s IPv6 Home Network Prefix. Only the first Prefix Length bytes are valid for the mobile node’s IPv6 Home Network Prefix. The rest of the bytes MUST be ignored.

4.6. Serving MAAR Option

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

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |     Length    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                               |
|                                                               |
|                     S-MAAR’s address                          |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type

iana-4.

Length

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

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

4.7. DLIF Link-local Address Option

A new DLIF Link-local Address option is defined for use with the Proxy Binding Update and Proxy Binding Acknowledgment messages exchanged between MAARs. This option is used for exchanging the link-local address of the DLIF to be configured on the serving MAAR so it resembles the DLIF configured on the P-MAAR.

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

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type        |    Length     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                               |
|                                                               |
|                  DLIF Link-local Address                      |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type

IANA-5.

Length

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

DLIF Link-local Address

A sixteen-byte field containing the link-local address of the logical interface.
4.8. DLIF Link-layer Address Option

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

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

```
+---------------+---------------+---------------+---------------+
| Type | Length | Reserved | DLIF Link-layer Address |
+---------------+---------------+---------------+---------------+
```

**Type**
IANA-6.

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

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

**DLIF Link-layer Address**
A variable length field containing the link-layer address of the logical interface to be configured on the S-MAAR.

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

This document defines new mobility options that require IANA actions: IANA-1 to IANA-6.

6. Security Considerations

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

7. Acknowledgments

The authors would like to thank Dirk von Hugo and John Kaippallimalil for the comments on this document. The authors would also like to thank Marco Liebsch, Dirk von Hugo, Alex Petrescu, Daniel Corujo, Akbar Rahman, Danny Moses, Xinpeng Wei and Satoru Matsushima for their comments and discussion on the documents [I-D.bernardos-dmm-distributed-anchoring] and [I-D.bernardos-dmm-pmip] on which the present document is based.

The authors would also like to thank Lyle Bertz and Danny Moses for their in-deep review of this document and their very valuable comments and suggestions.

8. References

8.1. Normative References


8.2. Informative References


Appendix A. Comparison with Requirement document

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

A.1. Distributed mobility management

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

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

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

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

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

A.3. IPv6 deployment

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

The DMM solution we propose targets IPv6 only.
A.4. Existing mobility protocols

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

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

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

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

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

A.6. Operation and management considerations

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

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

A.7. Security considerations

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

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

A.8. Multicast considerations

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

This solution in its current version does not specify any support for
multicast traffic.

Appendix B. Implementation experience

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

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

Tunnel management is hence a relevant technical aspect, as multiple
tunnels are established by a single MAAR, which keeps their status
directly into the MN’s BCE. Indeed, from the implementation
experience it was chosen to create an ancillary data structure as
field within a BCE: the data structure is called "MAAR list" and
stores the previous MAARs’ address and the corresponding prefix(es)
assigned for the MN. Only the CMD and the serving MAAR store this
data structure, because the CMD maintains the global MN’s mobility
session formed during the MN’s roaming within the domain, and the
serving MAAR needs to know which previous MAARs were visited, the
prefix(es) they assigned and the tunnels established with them.
Conversely, a previous MAAR only needs to know which is the current
Serving MAAR and establish a single tunnel with it. For this reason,
a MAAR that receives a PBU from the CMD (meaning that the MN attached
to another MAAR), first sets up the routing state for the MN’s
In order to have the MN totally unaware of the changes in the access link, all MAARs implement the Distributed Logical Interface (DLIF) concept. Moreover, it should be noted that the protocols designed in the document work only at the network layer to handle the MNs joining or leaving the domain. This should guarantee a certain independency to a particular access technology. The implementation reflects this reasoning, but we argue that an interaction with lower layers produces a more effective attachment and detachment detection, therefore improving the performance, also regarding de-registration mechanisms.

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

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

Authors’ Addresses

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

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

This document shows the applicability of SRv6 (Segment Routing IPv6) to the user-plane of mobile networks. The network programming nature of SRv6 accomplish mobile user-plane functions in a simple manner. The statelessness of SRv6 and its ability to control both service layer path and underlying transport can be beneficial to the mobile user-plane, providing flexibility and SLA control for various applications. This document describes the SRv6 mobile user plane behavior and defines the SID functions for that.
Table of Contents

1. Introduction .................................................. 3
2. Conventions and Terminology .................................. 3
   2.1. Terminology .............................................. 3
   2.2. Conventions .............................................. 4
   2.3. Predefined SRv6 Functions ............................... 4
3. Motivation ................................................... 5
4. A 3GPP Reference Architecture ................................. 6
5. User-plane behaviors ........................................... 7
   5.1. Traditional mode ......................................... 7
   5.1.1. Packet flow – Uplink ..................................... 8
   5.1.2. Packet flow – Downlink ................................... 8
   5.1.3. IPv6 user-traffic ....................................... 9
   5.2. Enhanced Mode ........................................... 9
   5.2.1. Packet flow – Uplink ..................................... 10
   5.2.2. Packet flow – Downlink ................................... 11
   5.2.3. IPv6 user-traffic ....................................... 11
   5.3. Enhanced mode with unchanged gNB GTP behavior ......... 11
   5.3.1. Interworking with IPv6 GTP ............................ 12
   5.3.2. Interworking with IPv4 GTP ............................ 15
   5.3.3. Extensions to the interworking mechanisms .......... 17
6. SRv6 SID Mobility Functions ................................ 18
   6.1. Args.Mob.Session ......................................... 18
   6.2. End.MAP .................................................. 18
   6.5. End.M.GTP4.E ............................................ 20
   6.6. T.M.Tmap ................................................ 20
   6.7. End.Limit: Rate Limiting function ...................... 21
7. SRv6 supported 3GPP PDU session types ...................... 22
8. Network Slicing Considerations ............................... 22
9. Control Plane Considerations ................................ 22
1. Introduction

In mobile networks, mobility management systems provide connectivity while mobile nodes move. While the control-plane of the system signals movements of a mobile node, the user-plane establishes a tunnel between the mobile node and its anchor node over IP-based backhaul and core networks.

This document shows the applicability of SRv6 (Segment Routing IPv6) to those mobile networks. SRv6 provides source routing to networks so that operators can explicitly indicate a route for the packets to and from the mobile node. SRv6 endpoint nodes serve as the anchors 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].

2.1. Terminology

- AMBR: Aggregate Maximum Bit Rate
- APN: Access Point Name (commonly used to identify a network or class of service)
- BSID: SR Binding SID [RFC8402]
- CNF: Cloud-native Network Function
- gNB: gNodeB
- NH: The IPv6 next-header field.
- NFV: Network Function Virtualization
- PDU: Packet Data Unit
- Session: TBD...
- SID: A Segment Identifier which represents a specific segment in a segment routing domain.
- SRH: The Segment Routing Header.
  [I-D.ietf-6man-segment-routing-header]
- TEID: Tunnel Endpoint Identifier
2.2. Conventions

- **NH=SRH** means that NH is 43 with routing type 4.
- Multiple SRHs may be present inside each packet, but they must follow each other. The next-header field of each SRH, except the last one, must be NH-SRH (43 type 4).
- For simplicity, no other extension headers are shown except the SRH.
- The SID type used in this document is IPv6 address (also called SRv6 Segment or SRv6 SID).
- gNB::1 is an IPv6 address (SID) assigned to the gNB.
- U1::1 is an IPv6 address (SID) assigned to UPF1.
- U2::1 is an IPv6 address (SID) assigned to UPF2.
- U2:: is some other IPv6 address (SID) assigned to UPF2.
- 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 SA and DA respectively, and next-header SRH, with SID list <S1, S2, S3> with SegmentsLeft = SL
  - The payload of the packet is not represented.
- 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.
- 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.
- SRH[SL] can be different from the DA of the IPv6 header.

2.3. Predefined SRv6 Functions

The following functions are defined in [I-D.filsfils-spring-srv6-network-programming].

- **End.DT4** means to decapsulate and forward using a specific IPv4 table lookup.
o End.DT6 means to decapsulate and forward using a specific IPv6 table lookup.
o End.DX4 means to decapsulate the packet and forward through a particular outgoing interface -or set of OIFs- configured with the SID.
o End.DX6 means to decapsulate and forward through a particular outgoing interface -or set of OIFs- configured with the SID.
o End.DX2 means to decapsulate the L2 frame and forward through a particular outgoing interface -or set of OIFs- configured with the SID.
o End.T means to forward using a specific IPv6 table lookup.
o End.X means to forward through a link configured with the SID.
T.Encaps.Red means encapsulation without pushing SRH (resulting in "Reduced" packet size).
o PSP means Penultimate Segment Pop. The packet is subsequently forwarded without the popped SRH.

New SRv6 functions are defined in Section 6 to support the needs of the mobile user plane.

3. Motivation

Mobility networks are becoming 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.

The current architecture of mobile networks does not take into account the underlying transport. The user-plane is rigidly fragmented into radio access, core and service networks, connected by tunneling according to user-plane roles such as access and anchor nodes. These factors have made it difficult for the operator to optimize and operate the data-path.

In the meantime, applications have shifted to use IPv6, and network operators have started adopting IPv6 as their IP transport. SRv6, the IPv6 dataplane instantiation of Segment Routing [RFC8402], integrates both the application data-path and the underlying transport layer into a single protocol, allowing operators to optimize the network in a simplified manner and removing forwarding state from the network. It is also suitable for virtualized environments, VNF/CNF to VNF/CNF networking.

SRv6 specifies network-programming (see [I-D.filsfils-spring-srv6-network-programming]). Applied to mobility, SRv6 can provide the user-plane functions needed for mobility management. SRv6 takes advantage of underlying transport awareness and flexibility to improve mobility user-plane functions.
The use-cases for SRv6 mobility are discussed in [I-D.camarilloelmkaly-springdmm-srv6-mob-usecases].

4. A 3GPP Reference Architecture

This section presents a reference architecture and possible deployment scenarios.

Figure 1 shows a reference diagram from the 5G packet core architecture [TS.23501].

The user plane described in this document does not depend on any specific architecture. The 5G packet core architecture as shown is based on the latest 3GPP standards at the time of writing this draft. Other architectures can be seen in [I-D.gundavelli-dmm-mfa] and [WHITEPAPER-5G-UP].

This reference diagram does not depict a UPF that is only connected to N9 interfaces, although the description in this document also work for such UPFs.

Each session from an UE gets assigned to a UPF. Sometimes multiple UPFs may be used, providing richer service functions. A UE gets its IP address from the DHCP block of its UPF. The UPF advertises that
IP address block toward the Internet, ensuring that return traffic is routed to the right UPF.

5. User-plane behaviors

This section describes some mobile user-plane behaviors using SRv6.

In order to simplify the adoption of SRv6, we present two different "modes" that vary with respect to the use of SRv6. The first one is the "Traditional mode", which inherits the current 3GPP mobile user-plane. In this mode there is no change to mobility networks architecture, except that GTP-U [TS.29281] is replaced by SRv6.

The second mode is the "Enhanced mode". In this mode the SR policy contains SIDs for Traffic Engineering and VNFs, which results in effective end-to-end network slices.

In both, the Traditional and the Enhanced modes, we assume that the gNB as well as the UPFs are SR-aware (N3, N9 and -potentially- N6 interfaces are SRv6).

We introduce two mechanisms for interworking with legacy access networks (N3 interface is unmodified). In these document we introduce them applied to the Enhanced mode, although they could be used in combination with the Traditional mode as well.

One of these mechanisms is designed to interwork with legacy gNBs using GTP/IPv4. The second method is designed to interwork with legacy gNBs using GTP/IPv6.

This document uses SRv6 functions defined in [I-D.filsfils-spring-srv6-network-programming] as well as new SRv6 functions designed for the mobile user plane. The new SRv6 functions are detailed in Section 6.

5.1. Traditional mode

In the traditional mode, the existing mobile UPFs remain unchanged except for the use of SRv6 as the data plane instead of GTP-U. There is no impact to the rest of mobile system.

In existing 3GPP mobile networks, an UE session is mapped 1-for-1 with a specific GTP tunnel (TEID). This 1-for-1 mapping is mirrored here to replace GTP encapsulation with the SRv6 encapsulation, while not changing anything else. There will be a unique SRv6 SID associated with each UE session.
The traditional mode minimizes the changes required to the mobile system; it is a good starting point for forming a common basis.

Our example topology is shown in Figure 2. In traditional mode the gNB and the UPFs are SR-aware. In the descriptions of the uplink and downlink packet flow, A is an IPv6 address of the UE, and Z is an IPv6 address reachable within the Data Network DN. A new SRv6 function End.MAP, defined in Section 6.2, is used.

5.1.1. Packet flow – Uplink

The uplink packet flow is as follows:

UE_out : (A,Z)

when the UE packet arrives at the gNB, the gNB performs a T.Encaps.Red operation. 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. gNB obtains the SID U1::1 from the existing control plane (N2 interface).

When the packet arrives at UPF1, the SID U1::1 identifies a local End.MAP function. End.MAP replaces U1::1 by U2::1, that belongs to the next UPF (U2).

When the packet arrives at UPF2, the SID U2::1 corresponds to an End.DT function. UPF2 decapsulates the packet, performs a lookup in a specific table associated with that mobile network and forwards the packet toward the data network (DN).

5.1.2. Packet flow – Downlink

The downlink packet flow is as follows:
When the packet arrives at the UPF2, the UPF2 maps that flow into a UE session. This UE session is associated with the segment endpoint \(<U1::1>\). UPF2 performs a T.Encaps.Red 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 to the next anchoring point and replaces U1::1 by gNB::1, that belongs to the next hop.

Upon packet arrival on gNB, the SID gNB::1 corresponds to an End.DX4 or End.DX6 function. The gNB decapsulates the packet, removing the IPv6 header and all its extensions headers, and forwards the traffic toward 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 [I-D.voyer-6man-extension-header-insertion]. The main benefit is a lower overhead (40B less). 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

Enhanced mode improves scalability, traffic steering and service programming [I-D.xuclad-spring-sr-service-programming], thanks to the use of multiple SIDs, instead of a single SID as done in the Traditional mode.

The main difference is that the SR policy MAY include SIDs for traffic engineering and service programming in addition to the UPFs SIDs.

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 using a mechanism like PCEP, DNS-lookup, small augment for LISP control-plane, etc.
Note that the SIDs MAY use the arguments Args.Mob.Session if required by the UPFs.

Figure 3 shows an Enhanced mode topology. In the Enhanced mode, the gNB and the UPF are SR-aware. The Figure shows two service segments, S1 and C1. S1 represents a VNF in the network, and C1 represents a constraint path on a router requiring Traffic Engineering. S1 and C1 belong to the underlay and don’t have an N4 interface, so they are not considered UPFs.

Figure 3: Enhanced mode - Example topology

5.2.1. Packet flow - Uplink

The uplink packet flow is as follows:

<table>
<thead>
<tr>
<th>UE_out</th>
<th>(A,Z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>gNB_out</td>
<td>(gNB, S1) (U2::1, C1; SL=2) (A,Z)</td>
</tr>
<tr>
<td>S1_out</td>
<td>(gNB, C1) (U2::1, C1; SL=1) (A,Z)</td>
</tr>
<tr>
<td>C1_out</td>
<td>(gNB, U2::1) (A,Z)</td>
</tr>
<tr>
<td>UPF2_out</td>
<td>(A,Z)</td>
</tr>
</tbody>
</table>

UE sends its packet (A,Z) on a specific bearer to its gNB. gNB’s control plane associates that session from the UE(A) with the IPv6 address B and GTP TEID T. gNB’s control plane does a lookup on B to find the related SID list <S1, C1, U2::1>.

When gNB transmits the packet, it contains all the segments of the SR policy. The SR policy can include segments for traffic engineering (C1) and for service programming (S1).

Nodes S1 and C1 perform their related Endpoint functionality and forward the packet.

When the packet arrives at UPF2, the active segment (U2::1) is an End.DT4/6 which performs the decapsulation (removing the IPv6 header with all its extension headers) and forwards toward the data network.
5.2.2. Packet flow - Downlink

The downlink packet flow is as follows:

UPF2_in : (Z,A) \rightarrow \text{UPF2 maps the flow w/ SID list \langle C1, S1, gNB \rangle}

UPF2_out: (U2::1, C1)(gNB, S1; SL=2)(Z,A) \rightarrow \text{T.Encaps.Red}

C1_out : (U2::1, S1)(gNB, S1; SL=1)(Z,A) \rightarrow \text{PSP}

S1_out : (U2::1, gNB)(Z,A) \rightarrow \text{End.DX4 or End.DX6}

gNB_out : (Z,A)

When the packet arrives at the UPF2, the UPF2 maps that particular flow into a UE session. This UE session is associated with the policy \langle C1, S1, gNB \rangle. The UPF2 performs a T.Encaps.Red 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 at the gNB, the IPv6 DA corresponds to an End.DX4 or End.DX6 (depending on the underlying traffic). The gNB decapsulates the packet, removing the IPv6 header and all its extensions headers and forwards the traffic toward 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

This section describes two mechanisms for interworking with legacy gNBs that still use GTP: one for IPv4, the other for IPv6.

In the interworking scenarios as illustrated in Figure 4, gNB does not support SrV6. gNB supports GTP encapsulation over IPv4 or IPv6. To achieve interworking, a SR Gateway (SRGW-UPF1) entity is added. The SRGW maps the GTP traffic into SrV6.

The SRGW is not an anchor point, and maintains very little state. For this reason, both IPv4 and IPv6 methods scale to millions of UEs.
5.3.1. Interworking with IPv6 GTP

In this interworking mode the gNB uses GTP over IPv6 via the N3 interface.

Key points:

- The gNB is unchanged (control-plane or user-plane) and encapsulates into GTP (N3 interface is not modified).
- The 5G Control-Plane (N2 interface) is unmodified; one IPv6 address is needed (i.e. a BSID at the SRGW).
- The SRGW removes GTP, finds the SID list related to DA, and adds 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; using Enhanced mode results in fewer SR policies on this node. A SR policy can be shared across UEs.
- When a packet from the UE leaves the gNB, it is SR-routed. This simplifies network slicing [I-D.hegdeppsenak-isis-sr-flex-algo].
- In the uplink, the IPv6 DA BSID steers traffic into an SR policy when it arrives at the SRGW-UPF1.

An example topology is shown in Figure 5. In this mode the gNB is an unmodified gNB using IPv6/GTP. The UPFs are SR-aware. As before, the SRGW maps IPv6/GTP traffic to SRv6.

S1 and C1 are two service segments. S1 represents a VNF in the network, and C1 represents a router configured for Traffic Engineering.
5.3.1.1. Packet flow - Uplink

The uplink packet flow is as follows:

**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 toward the gNB on a specific bearer for that session. The gNB, which is unmodified, encapsulates the packet into 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 at the SRGW. Hence the packet is routed to the SRGW.

When the packet arrives at the SRGW, the SRGW identifies B as an End.M.GTP6.D Binding SID (see Section 6.3). Hence, the SRGW removes the IPv6, UDP and GTP headers, and pushes an IPv6 header with its own SRH containing the SIDs bound to the SR policy associated with this BindingSID. There is one instance of the End.M.GTP6.D SID per PDU type.

S1 and C1 perform their related Endpoint functionality and forward the packet.

When the packet arrives at UPF2, the active segment is (U2::1) which is bound to End.DT4/6. UPF2 then decapsulates (removing the outer IPv6 header with all its extension headers) and forwards the packet toward the data network.
5.3.1.2. Packet flow - Downlink

The downlink packet flow is as follows:

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 table associated to A and finds the SID list <C1, S1, SRGW::TEID, gNB>. The UPF2 performs a T.Encaps.Red operation, encapsulating the packet into a new IPv6 header with its corresponding SRH.

C1 and S1 perform their related Endpoint processing.

Once the packet arrives at the SRGW, the SRGW identifies the active SID as an End.M.GTP6.E function. The SRGW removes the IPv6 header and all its extensions headers. The SRGW generates new 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 an argument of the received End.M.GTP6.E SID. The SRGW pushes the headers to the packet and forwards the packet toward the gNB. There is one instance of the End.M.GTP6.E SID per PDU type.

Once the packet arrives at 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 inserted by the UPF2. The UPF2 must have the UE states since it is the UE’s session anchor point.

For the uplink traffic, the state at the SRGW does not necessarily need to be unique per UE session; the state state can be shared among UEs. This enables much more scalable SRGW deployments compared to a solution holding millions of states, one or more per UE.
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 lower overhead.

5.3.2. Interworking with IPv4 GTP

In this interworking mode the gNB uses GTP over IPv4 in the N3 interface

Key points:

- The gNB is unchanged and encapsulates packets into GTP (the N3 interface is not modified).
- In the uplink, traffic is classified by SRGW’s Uplink Classifier and steered into an SR policy. The SRGW is a UPF1 functionality and can coexist with UPF1’s Uplink Classifier functionality.
- SRGW removes GTP, finds the SID list related to DA, and adds a SRH with the SID list.

An example topology is shown in Figure 6. In this mode the gNB is an unmodified gNB using IPv4/GTP. The UPFs are SR-aware. As before, the SRGW maps the IPv4/GTP traffic to SRv6.

S1 and C1 are two service segment endpoints. S1 represents a VNF in the network, and C1 represents a router configured for Traffic Engineering.

5.3.2.1. Packet flow - Uplink

The uplink packet flow is as follows:

```
        +----+         +----+         +----+         +----+
        |IPv4/GTP|        |S1|        |S1|        |DN|
        +----+         +----+         +----+         +----+
  |UE|--|gNB|--|SRv6|--|SRv6|--|C1|--|UPF2|--|DN|
  GTP +-----+         +-----+         +-----+         +-----+
  \    |    |    \    |    |    |    \    |
   \  |    |    \  |    |    \  |    |    \  |
      +-----+         +-----+         +-----+         +-----+
     |SR Gateway|       |SR Gateway|       |SR Gateway|       |SR Gateway|
```

Figure 6: Enhanced mode with unchanged gNB IPv4/GTP behavior
The UE sends a packet destined to Z toward 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 at the SRGW for UPF1, the SRGW has an 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.

S1 and C1 perform their related Endpoint functionality and forward the packet.

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 its extension headers) and forwards toward the data network.

5.3.2.2. Packet flow - Downlink

The downlink packet flow is as follows:

\[
gNB\_out : (gNB, B)(GTP: TEID T)(A,Z) \rightarrow Interface N3 unchanged IPv4/GTP
SRGW\_out: (SRGW, S1)(U2::1, C1; SL=2)(A,Z) \rightarrow T.M.Tmap function
S1\_out : (SRGW, C1)(U2::1, C1; SL=1)(A,Z) \rightarrow PSP
C1\_out : (SRGW, U2::1) (A,Z) \rightarrow End.DT4 or End.DT6
UPF2\_out: (A,Z)
\]

When a packet destined to A arrives at the UPF2, the UPF2 performs a lookup in the table associated to A and finds the SID list <C1, S1, SRGW::SA:DA:TEID>. The UPF2 performs a T.Encaps.Red 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 at the SRGW, the SRGW identifies the active SID as an End.M.GTP4.E function. The SRGW removes the IPv6 header and all its extensions headers. The SRGW generates an IPv4, UDP and GTP headers. The IPv4 SA and DA are received as 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 toward the gNB.

When the packet arrives at 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 inserted by the UPF. The UPF must have this UE-base state anyway (since it is its anchor point).

For the uplink traffic, the state at the SRGW is dedicated on a per UE/session basis according to an Uplink Classifier. There is state for steering the different sessions on a SR policies. However, 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. 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. Args.Mob.Session

Args.Mob.Session provide per-session information for charging, buffering and lawful intercept (among others) required by some mobile nodes. The Args.Mob.Session argument format is used in combination with End.Map, End.DT and End.DX functions. Note that proposed format is applicable for 5G networks, while similar formats could be proposed for legacy networks.

<table>
<thead>
<tr>
<th>QFI</th>
<th>R</th>
<th>U</th>
<th>PDU Session ID</th>
</tr>
</thead>
</table>

Args.Mob.Session format

- QFI: QoS Flow Identifier [TS.38415]
- R: Reflective QoS Indication [TS.23501]. This parameter indicates the activation of reflective QoS towards the UE for the transferred packet. Reflective QoS enables the UE to map UL User Plane traffic to QoS Flows without SMF provided QoS rules.
- U: Unused and for future use. MUST be 0 on transmission and ignored on receipt.
- PDU Session ID: Identifier of PDU Session. The GTP-U equivalent is TEID.

Since the SRv6 function is likely NOT to be instantiated per PDU session, Args.Mob.Session helps the UPF to perform the functions which require per QFI and/or per PDU Session granularity.

6.2. End.MAP

The "Endpoint function with SID mapping" function (End.MAP for short) is used in several scenarios. Particularly in mobility, End.MAP is used in the UPFs for the PDU Session anchor functionality.

When a SR node N receives a packet destined to S and S is a local End.MAP SID, N does the following:
1. look up the IPv6 DA in the mapping table
2. update the IPv6 DA with the new mapped SID  ;; Note 1
3. IF segment_list > 1
4. insert a new SRH
5. forward according to the new mapped SID
6. ELSE
7. Drop the packet

Note 1: The SID in the SRH is NOT modified.

6.3. End.M.GTP6.D

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 toward from the legacy gNB using IPv6/GTP. Suppose, for example, 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 IPv6, 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 S1 segment of the SRv6 Policy
7. ELSE
8. Drop the packet


The "Endpoint function with encapsulation for IPv6/GTP tunnel" function (End.M.GTP6.E for short) is used in interworking scenario for the downlink toward the legacy gNB using IPv6/GTP.

The End.M.GTP6.E function has a 32-bit argument space which is used to provide 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 the following:
1. IF NH=SRH & SL = 1 THEN ;; Note 1
2. decrement SL
3. store SRH[SL] in variable new_DA
4. store TEID in variable new_TEID ;; Note 2
5. pop IP header and all its 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

Note 1: An End.M.GTP6.E SID MUST always be the penultimate SID.

Note 2: TEID is extracted from the argument space of the current SID.

6.5. End.M.GTP4.E

The "Endpoint function with encapsulation for IPv4/GTP tunnel" function (End.M.GTP4.E 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. store SRH[0] in buffer S
3. pop the IPv6 header and its extension headers
4. push UDP/GTP headers with GTP TEID from S
5. push outer IPv4 header with SA, DA from S
6. ELSE
7. Drop the packet

S has the following format:

```
+---------------------+-------+-------+-------+
| SRGW-IPv6-LOC-FUNC  |IPv4DA |IPv4SA |TUN-ID |
|---------------------+-------+-------+-------+
```

End.M.GTP4.E SID Encoding

6.6. T.M.Tmap

The "Transit with tunnel decapsulation and map to an SRv6 policy" function (T.M.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 Payload == UDP/GTP THEN
2. pop the outer IPv4 header and UDP/GTP headers
3. copy IPv4 DA, SA, TUN-ID to form SID B
4. encapsulate the packet into a new IPv6 header
5. set the IPv6 DA = B
6. forward along the shortest path to B
7. ELSE
8. Drop the packet

B has the following format:

+---------------------+-------+-------+-------+
| SRGW-IPv6-LOC-FUNC  |IPv4DA |IPv4SA |TUN-ID |
+---------------------+-------+-------+-------+
  128-a-b-c          a      b       c

End.M.GTP4.E SID Encoding

The SID B is an SRv6 BindingSID instantiated at the first UPF (U1). A static format is used for this Binding SIDs in order to remove state from the SRGW.

6.7. End.Limit: Rate Limiting function

The mobile user-plane requires a rate-limit feature. For this purpose, we define a new function "End.Limit". The "End.Limit" function encodes in its arguments the rate limiting parameter that should be applied to this packet. Multiple flows of packets should have the same group identifier in the SID when those flows are in a same AMBR group. The encoding format of the rate limit segment SID is as follows:

+---------------------+---------------------+---------------------+
| LOC+FUNC rate-limit | group-id | limit-rate |
+---------------------+---------------------+---------------------+
  128-i-j          i      j

End.Limit: Rate limiting function argument format

If the limit-rate bits are set to zero, the node should not do rate limiting unless static configuration or control-plane sets the limit rate associated to the SID.
7. SRv6 supported 3GPP PDU session types

The 3GPP [TS.23501] defines the following PDU session types:

- IPv4
- IPv6
- IPv4v6
- Ethernet
- Unstructured

SRv6 supports all the 3GPP PDU session types without any protocol overhead by using the corresponding SRv6 functions (End.DX4, End.DT4 for IPv4 PDU sessions; End.DX6, End.DT6, End.T for IPv6 PDU sessions; End.DT46 for IPv4v6 PDU sessions; End.DX2 for L2 PDU sessions; End.DX2 for Unstructured PDU sessions).

8. 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.

[I-D.filsfils-spring-segment-routing-policy] describes a solution to build basic network slices with SR. Depending on the requirements, these slices can be further refined by adopting the mechanisms from:

- IGP Flex-Algo [I-D.hegdeppsenak-isis-sr-flex-algo]
- Inter-Domain policies
  [I-D.ietf-spring-segment-routing-central-epe]

Furthermore, these can be combined with ODN/AS
[I-D.filsfils-spring-segment-routing-policy] for automated slice provisioning and traffic steering.

Further details on how these tools can be used to create end to end network slices are documented in
[I-D.ali-spring-network-slicing-building-blocks].

9. Control Plane Considerations

This document focuses on user-plane behavior and its independence from the control plane.

The control plane could be the current 3GPP-defined control plane with slight modifications to the N4 interface [TS.29244].

Alternatively, SRv6 could be used in conjunction with a new mobility control plane as described in LISP [I-D.rodrigueznatal-lisp-srv6],
hICN [I-D.auge-dmm-hicn-mobility-deployment-options], MFA
[I-D.gundavelli-dmm-mfa] or in conjunction with FPC
[I-D.ietf-dmm-fpc-cpdp]. The analysis of new mobility control-planes
and its applicability to SRv6 is out of the scope of this document.

Section 11 allocates SRv6 endpoint function types for the new
functions defined in this document. Control-plane protocols are
expected to use these function type codes to signal each function.

SRv6’s network programming nature allows a flexible and dynamic UPF
placement.

10. Security Considerations

TBD

11. IANA Considerations

IANA is requested 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 values:

<table>
<thead>
<tr>
<th>Value/Range</th>
<th>Hex</th>
<th>Endpoint function</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBA</td>
<td>TBA</td>
<td>End.MAP</td>
<td>[This.ID]</td>
</tr>
<tr>
<td>TBA</td>
<td>TBA</td>
<td>End.M.GTP6.D</td>
<td>[This.ID]</td>
</tr>
<tr>
<td>TBA</td>
<td>TBA</td>
<td>End.M.GTP6.E</td>
<td>[This.ID]</td>
</tr>
<tr>
<td>TBA</td>
<td>TBA</td>
<td>End.M.GTP4.E</td>
<td>[This.ID]</td>
</tr>
<tr>
<td>TBA</td>
<td>TBA</td>
<td>End.Limit</td>
<td>[This.ID]</td>
</tr>
</tbody>
</table>

Table 1: SRv6 Mobile User-plane Endpoint Types

12. Acknowledgements

The authors would like to thank Daisuke Yokota, Bart Peirens,
Ryokichi Onishi, Kentaro Ebisawa, Peter Bosch, Darren Dukes, Francois
Clad, Sri Gundavelli, Sridhar Bhaskaran, Arashmid Akhavain, Ravi
Shekhar and Aeneas Dodd-Noble for their useful comments of this work.

13. Contributors

Kentaro Ebisawa
Ponto Networks
Japan

14. References

14.1. Normative References

[I-D.filsfils-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., Hegde, S.,
daniel.voyer@bell.ca, d., Lin, S., bogdanov@google.com,
b., Krol, P., Horneffer, M., Steinberg, D., Decraene, B.,
Litkowski, S., Mattes, P., Ali, Z., Talaulikar, K., Liste,
J., Clad, F., and K. Raza, "Segment Routing Policy
Architecture", draft-filsfils-spring-segment-routing-
policy-06 (work in progress), May 2018.

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J.,
daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6
Network Programming", draft-filsfils-spring-srv6-network-
programming-07 (work in progress), February 2019.

[I-D.ietf-6man-segment-routing-header]
Filsfils, C., Previdi, S., Leddy, J., Matsushima, S., and
b. daniel.voyer@bell.ca, "IPv6 Segment Routing Header
(SRH)", draft-ietf-6man-segment-routing-header-16 (work in
progress), February 2019.

[I-D.voyer-6man-extension-header-insertion]
daniel.voyer@bell.ca, d., Leddy, J., Filsfils, C., Dukes,
D., Previdi, S., and S. Matsushima, "Insertion of IPv6
Segment Routing Headers in a Controlled Domain", draft-
voyer-6man-extension-header-insertion-05 (work in
progress), January 2019.

[RFC2119]  Bradner, S., "Key words for use in RFCs to Indicate
        Requirement Levels", BCP 14, RFC 2119,
        DOI 10.17487/RFC2119, March 1997,

        Decraene, B., Litkowski, S., and R. Shakir,
        "Segment Routing Architecture", RFC 8402,
        DOI 10.17487/RFC8402, July 2018,
14.2. Informative References

[I-D.ali-spring-network-slicing-building-blocks]

[I-D.auge-dmm-hicn-mobility-deployment-options]

[I-D.camarillo-dmm-srv6-mobile-pocs]

[I-D.camarilloelmalky-springdmm-srv6-mob-usecases]

[I-D.gundavelli-dmm-mfa]

[I-D.hegdeppsenak-isis-sr-flex-algo]

[I-D.ietf-dmm-fpc-cpdp]

[I-D.ietf-spring-segment-routing-central-epe]
Appendix A. Implementations

This document introduces new SRv6 functions. These functions have an open-source P4 implementation available in <https://github.com/ebiken/p4srv6>.

There are also implementations in M-CORD NGIC and Open Air Interface (OAI). Further details can be found in [I-D.camarillo-dmm-srv6-mobile-pocs].

Appendix B. Changes from revision 02 to revision 03

This section lists the changes between draft-ietf-dmm-srv6-mobile-uplane revisions ...-02 and ...-03.

- Added new terminology section for abbreviations.
- Added new terminology section for predefined SRv6 functions.
- Made terminology section for conventions used in the document.
o Renamed "Basic" mode to be called "Traditional" mode.
o Renamed "Aggregate" mode to be called "Enhanced" mode.
o Added new Args.Mob.Session format to supply QFI, RQI indication and PDU Session ID.
o Modified End.MAP function to define the SID argument format and support more than one SID
o Added missing references.
o Editorial updates to improve readability.

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
A Simple BGP-based Mobile Routing System for the Aeronautical Telecommunications Network

Abstract

The International Civil Aviation Organization (ICAO) is investigating mobile routing solutions for a worldwide Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS). The ATN/IPS will eventually replace existing communication services with an IPv6-based service supporting pervasive Air Traffic Management (ATM) for Air Traffic Controllers (ATC), Airline Operations Controllers (AOC), and all commercial aircraft worldwide. This informational document describes a simple and extensible mobile routing service based on industry-standard BGP to address the ATN/IPS requirements.

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 November 15, 2019.
1. Introduction

The worldwide Air Traffic Management (ATM) system today uses a service known as Aeronautical Telecommunications Network based on Open Systems Interconnection (ATN/OSI). The service is used to augment controller to pilot voice communications with rudimentary short text command and control messages. The service has seen successful deployment in a limited set of worldwide ATM domains.

The International Civil Aviation Organization (ICAO) is now undertaking the development of a next-generation replacement for ATN/OSI known as Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS). ATN/IPS will eventually provide an
IPv6-based [RFC8200] service supporting pervasive ATM for Air Traffic Controllers (ATC), Airline Operations Controllers (AOC), and all commercial aircraft worldwide. As part of the ATN/IPS undertaking, a new mobile routing service will be needed. This document presents an approach based on the Border Gateway Protocol (BGP) [RFC4271].

Aircraft communicate via wireless aviation data links that typically support much lower data rates than terrestrial wireless and wired-line communications. For example, some Very High Frequency (VHF)-based data links only support data rates on the order of 32Kbps and an emerging L-Band data link that is expected to play a key role in future aeronautical communications only supports rates on the order of 1Mbps. Although satellite data links can provide much higher data rates during optimal conditions, like any other aviation data link they are subject to errors, delay, disruption, signal intermittence, degradation due to atmospheric conditions, etc. The well-connected ground domain ATN/IPS network should therefore treat each safety-of-flight critical packet produced by (or destined to) an aircraft as a precious commodity and strive for an optimized service that provides the highest possible degree of reliability.

The ATN/IPS is an IPv6-based overlay network configured over one or more Internetworking underlays ("INETs") maintained by aeronautical network service providers such as ARINC, SITA and Inmarsat. Each INET comprises one or more "partitions" where all nodes within a partition can exchange packets with all other nodes, i.e., the partition is connected internally. There is no requirement that any two INET partitions use the same IP protocol version nor have consistent IP addressing plans in comparison with other partitions. Instead, the ATN/IPS IPv6 overlay sees each partition as a "segment" of a link-layer topology manifested through a (virtual) bridging service known as "Spanning Partitioned Aeronautical Networks (SPAN)". Further discussion of the SPAN is found in the following sections of this document, with reference to [I-D.templin-intarea-6706bis].

The ATN/IPS further assumes that each aircraft will receive an IPv6 Mobile Network Prefix (MNP) that accompanies the aircraft wherever it travels. ICAO is further proposing to assign each aircraft an entire /56 MNP for numbering its on-board networks. ATCs and AOCs will likewise receive IPv6 prefixes, but they would typically appear in static (not mobile) deployments such as air traffic control towers, airline headquarters, etc. Throughout the rest of this document, we therefore use the term "MNP" when discussing an IPv6 prefix that is delegated to any ATN/IPS end system, including ATCs, AOCs, and aircraft. We also use the term Mobility Service Prefix (MSP) to refer to an aggregated prefix assigned to the ATN/IPS by an Internet assigned numbers authority, and from which all MNPs are delegated.
Connexion By Boeing [CBB] was an early aviation mobile routing service based on dynamic updates in the global public Internet BGP routing system. Practical experience with the approach has shown that frequent injections and withdrawals of MNPs in the Internet routing system can result in excessive BGP update messaging, slow routing table convergence times, and extended outages when no route is available. This is due to both conservative default BGP protocol timing parameters (see Section 6) and the complex peering interconnections of BGP routers within the global Internet infrastructure. The situation is further exacerbated by frequent aircraft mobility events that each result in BGP updates that must be propagated to all BGP routers in the Internet that carry a full routing table.

We therefore consider an approach using a BGP overlay network routing system where a private BGP routing protocol instance is maintained between ATN/IPS Autonomous System (AS) Border Routers (ASBRs). The private BGP instance does not interact with the native BGP routing systems in underlying INETs, and BGP updates are unidirectional from "stub" ASBRs (s-ASBRs) to a small set of "core" ASBRs (c-ASBRs) in a hub-and-spokes topology. No extensions to the BGP protocol are necessary.

The s-ASBRs for each stub AS connect to a small number of c-ASBRs via dedicated high speed links and/or tunnels across the INET using industry-standard encapsulations (e.g., Generic Routing Encapsulation (GRE) [RFC2784], IPsec [RFC4301], etc.). In particular, tunneling must be used when neighboring ASBRs are separated by multiple INET hops.

The s-ASBRs engage in external BGP (eBGP) peerings with their respective c-ASBRs, and only maintain routing table entries for the MNPs currently active within the stub AS. The s-ASBRs send BGP updates for MNP injections or withdrawals to c-ASBRs but do not receive any BGP updates from c-ASBRs. Instead, the s-ASBRs maintain default routes with their c-ASBRs as the next hop, and therefore hold only partial topology information.

The c-ASBRs connect to other c-ASBRs within the same partition using internal BGP (iBGP) peerings over which they collaboratively maintain a full routing table for all active MNPs currently in service within the partition. Therefore, only the c-ASBRs maintain a full BGP routing table and never send any BGP updates to s-ASBRs. This simple routing model therefore greatly reduces the number of BGP updates that need to be synchronized among peers, and the number is reduced
further still when intradomain routing changes within stub ASes are processed within the AS instead of being propagated to the core. BGP Route Reflectors (RRs) [RFC4456] can also be used to support increased scaling properties.

When there are multiple INET partitions, the c-ASBRs of each partition use eBGP to peer with the c-ASBRs of other partitions so that the full set of MNPs for all partitions are known globally among all of the c-ASBRs. Each c/s-ASBR further configures a "SPAN address" which is taken from a global or unique-local IPv6 "SPAN prefix" assigned to each partition, as well as static forwarding table entries for all other prefixes in the SPAN. The SPAN addresses are used for nested encapsulation where the inner IPv6 packet is encapsulated in a SPAN header which is then encapsulated in an IP header specific to the INET partition.

The remainder of this document discusses the proposed BGP-based ATN/IPS mobile routing service.

2. Terminology

The terms Autonomous System (AS) and Autonomous System Border Router (ASBR) are the same as defined in [RFC4271].

The following terms are defined for the purposes of this document:

**Air Traffic Management (ATM)**

The worldwide service for coordinating safe aviation operations.

**Air Traffic Controller (ATC)**

A government agent responsible for coordinating with aircraft within a defined operational region via voice and/or data Command and Control messaging.

**Airline Operations Controller (AOC)**

An airline agent responsible for tracking and coordinating with aircraft within their fleet.

**Aeronautical Telecommunications Network with Internet Protocol Services (ATN/IPS)**

A future aviation network for ATCs and AOCs to coordinate with all aircraft operating worldwide. The ATN/IPS will be an IPv6-based overlay network service that connects access networks via tunneling over one or more Internetworking underlays.

**Internetworking underlay ("INET")**

A wide-area network that supports overlay network tunneling and connects Radio Access Networks to the rest of the ATN/IPS.
Example INET service providers for civil aviation include ARINC, SITA and Inmarsat.

(Radio) Access Network ("ANET")
An aviation radio data link service provider’s network, including radio transmitters and receivers as well as supporting ground-domain infrastructure needed to convey a customer’s data packets to outside INETs. The term ANET is intended in the same spirit as for radio-based Internet service provider networks (e.g., cellular operators), but can also refer to ground-domain networks that connect AOCs and ATCs.

partition (or "segment")
A fully-connected internal subnetwork of an INET in which all nodes can communicate with all other nodes within the same partition using the same IP protocol version and addressing plan. Each INET consists of one or more partitions.

Spanning Partitioned Aeronautical Networks (SPAN)
A virtual layer 2 bridging service that presents a unified link view to the ATN/IPS overlay even though the underlay may consist of multiple INET partitions. The SPAN is manifested through nested encapsulation in which IPv6 packets from the ATN/IPS are first encapsulated in SPAN headers which are then encapsulated in INET headers. In this way, packets sent from a source can be conveyed over the SPAN even though there may be many underlying INET partitions in the path to the destination.

SPAN Autonomous System
A "hub-of-hubs" autonomous system maintained through peerings between the core autonomous systems of different SPAN partitions.

Core Autonomous System Border Router (c-ASBR)
A BGP router located in the hub of the INET partition hub-and-spokes overlay network topology.

Core Autonomous System
The "hub" autonomous system maintained by all c-ASBRs within the same partition.

Stub Autonomous System Border Router (s-ASBR)
A BGP router configured as a spoke in the INET partition hub-and-spokes overlay network topology.

Stub Autonomous System
A logical grouping that includes all Clients currently associated with a given s-ASBR.
Client
An ATC, AOC or aircraft that connects to the ATN/IPS as a leaf node. The Client could be a singleton host, or a router that connects a mobile or fixed network.

Proxy
An ANET/INET border node that acts as a transparent intermediary between Clients and s-ASBRs. From the Client's perspective, the Proxy presents the appearance that the Client is communicating directly with the s-ASBR. From the s-ASBR’s perspective, the Proxy presents the appearance that the s-ASBR is communicating directly with the Client.

Mobile Network Prefix (MNP)
An IPv6 prefix that is delegated to any ATN/IPS end system, including ATCs, AOCs, and aircraft.

Mobility Service Prefix (MSP)
An aggregated prefix assigned to the ATN/IPS by an Internet assigned numbers authority, and from which all MNPs are delegated (e.g., up to $2^{32}$ IPv6 /56 MNPs could be delegated from a /24 MSP).

3. ATN/IPS Routing System

The ATN/IPS routing system comprises a private BGP instance coordinated in an overlay network via tunnels between neighboring ASBRs over one or more underlying INETs. The overlay does not interact with the underlying INET BGP routing systems, and only a small and unchanging set of MSPs are advertised externally instead of the full dynamically changing set of MNPs.

Within each INET partition, one or more s-ASBRs connect each stub AS to the INET partition core using a shared stub AS Number (ASN). Each s-ASBR further uses eBGP to peer with one or more c-ASBRs. All c-ASBRs are members of the INET partition core AS, and use a shared core ASN. Globally-unique public ASNs could be assigned, e.g., either according to the standard 16-bit ASN format or the 32-bit ASN scheme defined in [RFC6793].

The c-ASBRs use iBGP to maintain a synchronized consistent view of all active MNPs currently in service within the INET partition. Figure 1 below represents the reference INET partition deployment. (Note that the figure shows details for only two s-ASBRs (s-ASBR1 and s-ASBR2) due to space constraints, but the other s-ASBRs should be understood to have similar Stub AS, MNP and eBGP peering arrangements.) The solution described in this document is flexible enough to extend to these topologies.
In the reference deployment, each s-ASBR maintains routes for active MNPs that currently belong to its stub AS. In response to "Inter-domain" mobility events, each s-ASBR will dynamically announces new MNPs and withdraws departed MNPs in its eBGP updates to c-ASBRs. Since ATN/IPS end systems are expected to remain within the same stub AS for extended timeframes, however, intra-domain mobility events (such as an aircraft handing off between cell towers) are handled within the stub AS instead of being propagated as inter-domain eBGP updates.

Each c-ASBR configures a black-hole route for each of its MSPs. By black-holing the MSPs, the c-ASBR will maintain forwarding table
entries only for the MNPs that are currently active, and packets destined to all other MNPs will correctly incur ICMPv6 Destination Unreachable messages [RFC4443] due to the black hole route. (This is the same behavior as for ordinary BGP routers in the Internet when they receive packets for which there is no route available.) The c-ASBRs do not send eBGP updates for MNPs to s-ASBRs, but instead originate a default route. In this way, s-ASBRs have only partial topology knowledge (i.e., they know only about the active MNPs currently within their stub ASes) and they forward all other packets to c-ASBRs which have full topology knowledge.

The core ASes of each INET partition are joined together through external BGP peerings. The c-ASBRs of each partition establish external peerings with the c-ASBRs of other partitions to form a "core-of-cores" SPAN AS. The SPAN AS contains the global knowledge of all MNPs deployed worldwide, and supports ATN/IPS overlay communications between nodes located in different INET partitions by virtue of SPAN encapsulation. Figure 2 shows a reference SPAN topology.

```
+----------------+  +----------------+  +----------------+
|                |  |                |  |                |
| (::: Partition 1 ::)--|c-ASBR|---+        |
|                |  |                |  |                |
```

```
+----------------+  +----------------+  +----------------+
|                |  |                |  |                |
| (::: Partition 2 ::)--|c-ASBR|---+        |
|                |  |                |  |                |
```

```
+----------------+  +----------------+  +----------------+
|                |  |                |  |                |
| (::: Partition 3 ::)--|c-ASBR|---+        |
|                |  |                |  |                |
```

<- ATN/IPS Overlay Bridged by the SPAN AS ->

Figure 2: The SPAN
Scaling properties of this ATN/IPS routing system are limited by the number of BGP routes that can be carried by the c-ASBRs. A 2015 study showed that BGP routers in the global public Internet at that time carried more than 500K routes with linear growth and no signs of router resource exhaustion [BGP]. A more recent network emulation study also showed that a single c-ASBR can accommodate at least 1M dynamically changing BGP routes even on a lightweight virtual machine. Commercially-available high-performance dedicated router hardware can support many millions of routes.

Therefore, assuming each c-ASBR can carry 1M or more routes, this means that at least 1M ATN/IPS end system MNPs can be serviced by a single set of c-ASBRs and that number could be further increased by using RRs and/or more powerful routers. Another means of increasing scale would be to assign a different set of c-ASBRs for each set of MSPs. In that case, each s-ASBR still peers with one or more c-ASBRs from each set of c-ASBRs, but the s-ASBR institutes route filters so that it only sends BGP updates to the specific set of c-ASBRs that aggregate the MSP. In this way, each set of c-ASBRs maintains separate routing and forwarding tables so that scaling is distributed across multiple c-ASBR sets instead of concentrated in a single c-ASBR set. For example, a first c-ASBR set could aggregate an MSP segment A::/32, a second set could aggregate B::/32, a third could aggregate C::/32, etc. The union of all MSP segments would then constitute the collective MSP(s) for the entire ATN/IPS, with potential for supporting many millions of mobile networks or more.

In this way, each set of c-ASBRs services a specific set of MSPs, and each s-ASBR configures MSP-specific routes that list the correct set of c-ASBRs as next hops. This design also allows for natural incremental deployment, and can support initial medium-scale deployments followed by dynamic deployment of additional ATN/IPS infrastructure elements without disturbing the already-deployed base. For example, a few more c-ASBRs could be added if the MNP service demand ever outgrows the initial deployment. For larger-scale applications (such as unmanned air vehicles and terrestrial vehicles) even larger scales can be accommodated by adding more c-ASBRs.

4. ATN/IPS (Radio) Access Network (ANET) Model

(Radio) Access Networks (ANETs) connect end system Clients such as aircraft, ATCs, AOCs etc. to the ATN/IPS routing system. Clients may connect to multiple ANETs at once, for example, when they have both satellite and cellular data links activated simultaneously. Clients may further move between ANETs in a manner that is perceived as a network layer mobility event. Clients could therefore employ a multilink/mobility routing service such as those discussed in Section 7.
Clients register all of their active data link connections with their serving s-ASBRs as discussed in Section 3. Clients may connect to s-ASBRs either directly, or via a Proxy at the anet/inet boundary.

Figure 3 shows the ATN/IPS ANET model where Clients connect to ANETs via aviation data links. Clients register their ANET addresses with a nearby s-ASBR, where the registration process may be brokered by a Proxy at the edge of the ANET.

When a Client logs into an ANET it specifies a nearby s-ASBR that it has selected to connect to the ATN/IPS. (Selection of a nearby s-ASBR could be through consulting a geographically-keyed static host file, through a DNS lookup, through a network query response, etc.) The login process is transparently brokered by a Proxy at the border of the ANET, which then conveys the connection request to the s-ASBR via tunneling across the SPAN. The s-ASBR then registers the address of the Proxy as the address for the Client, and the Proxy forwards the s-ASBR’s reply to the Client. If the Client connects to multiple...
ANETs, the s-ASBR will register the addresses of all Proxies as addresses through which the Client can be reached.

The s-ASBR represents all of its active Clients as MNP routes in the ATN/IPS BGP routing system. The s-ASBR’s stub AS therefore consists of the set of all of its active Clients (i.e., the stub AS is a logical construct and not a physical construct). The s-ASBR injects the MNPs of its active Clients and withdraws the MNPs of its departed Clients via BGP updates to c-ASBRs, which further propagate the MNPs to other c-ASBRs within the SPAN AS. Since Clients are expected to remain associated with their current s-ASBR for extended periods, the level of MNP injections and withdrawals in the BGP routing system will be on the order of the numbers of network joins, leaves and s-ASBR handovers for aircraft operations (see: Section 6). It is important to observe that fine-grained events such as Client mobility and Quality of Service (QoS) signaling are coordinated only by Proxies and the Client’s current s-ASBRs, and do not involve other ASBRs in the routing system. In this way, intradomain routing changes within the stub AS are not propagated into the rest of the ATN/IPS BGP routing system.

5. ATN/IPS Route Optimization

ATN/IPS end systems will frequently need to communicate with correspondents associated with other s-ASBRs. In the BGP peering topology discussed in Section 3, this can initially only be accommodated by including multiple tunnel segments in the forwarding path. In many cases, it would be desirable to eliminate extraneous tunnel segments from this "dogleg" route so that packets can traverse a minimum number of tunneling hops across the SPAN. ATN/IPS end systems could therefore employ a route optimization service according to the mobility service employed (see: Section 7).

A route optimization example is shown in Figure 4 and Figure 5 below. In the first figure, multiple tunneled segments between Proxys and ASBRs are necessary to convey packets between Clients associated with different s-ASBRs. In the second figure, the optimized route tunnels packets directly between Proxys without involving the ASBRs.
Figure 4: Dogleg Route Before Optimization
6. BGP Protocol Considerations

The number of eBGP peering sessions that each c-ASBR must service is proportional to the number of s-ASBRs in its local partition. Network emulations with lightweight virtual machines have shown that a single c-ASBR can service at least 100 eBGP peerings from s-ASBRs that each advertise 10K MNP routes (i.e., 1M total). It is expected that robust c-ASBRs can service many more peerings than this—possibly by multiple orders of magnitude. But even assuming a conservative limit, the number of s-ASBRs could be increased by also increasing the number of c-ASBRs. Since c-ASBRs also peer with each other using iBGP, however, larger-scale c-ASBR deployments may need to employ an adjunct facility such as BGP Route Reflectors (RRs)[RFC4456].

The number of aircraft in operation at a given time worldwide is likely to be significantly less than 1M, but we will assume this...
number for a worst-case analysis. Assuming a worst-case average 1 hour flight profile from gate-to-gate with 10 service region transitions per flight, the entire system will need to service at most 10M BGP updates per hour (2778 updates per second). This number is within the realm of the peak BGP update messaging seen in the global public Internet today [BGP2]. Assuming a BGP update message size of 100 bytes (800 bits), the total amount of BGP control message traffic to a single c-ASBR will be less than 2.5 Mbps which is a nominal rate for modern data links.

Industry standard BGP routers provide configurable parameters with conservative default values. For example, the default hold time is 90 seconds, the default keepalive time is 1/3 of the hold time, and the default MinRouteAdvertisementInterval is 30 seconds for eBGP peers and 5 seconds for iBGP peers (see Section 10 of [RFC4271]). For the simple mobile routing system described herein, these parameters can and should be set to more aggressive values to support faster neighbor/link failure detection and faster routing protocol convergence times. For example, a hold time of 3 seconds and a MinRouteAdvertisementInterval of 0 seconds for both iBGP and eBGP.

Each c-ASBR will be using eBGP both in the ATN/IPS and the INET with the ATN/IPS unicast IPv6 routes resolving over INET routes. Consequently, c-ASBRs and potentially s-ASBRs will need to support separate local ASes for the two BGP routing domains and routing policy or assure routes are not propagated between the two BGP routing domains. From a conceptual and operational standpoint, the implementation should provide isolation between the two BGP routing domains (e.g., separate BGP instances).

7. Stub AS Mobile Routing Services

Stub ASes maintain intradomain routing information for mobile node clients, and are responsible for all localized mobility signaling without disturbing the BGP routing system. Clients can enlist the services of a candidate mobility service such as Mobile IPv6 (MIPv6) [RFC6275], LISP [I-D.ietf-lisp-rfc6830bis] and AERO [I-D.templin-intarea-6706bis] according to the service offered by the stub AS. Further details of mobile routing services are out of scope for this document.

8. Implementation Status

The BGP routing topology described in this document has been modeled in realistic network emulations showing that at least 1 million MNPs can be propagated to each c-ASBR even on lightweight virtual machines. No BGP routing protocol extensions need to be adopted.
9. IANA Considerations

This document does not introduce any IANA considerations.

10. Security Considerations

ATN/IPS ASBRs on the open Internet are susceptible to the same attack profiles as for any Internet nodes. For this reason, ASBRs should employ physical security and/or IP securing mechanisms such as IPsec [RFC4301], TLS [RFC5246], etc.

ATN/IPS ASBRs present targets for Distributed Denial of Service (DDoS) attacks. This concern is no different than for any node on the open Internet, where attackers could send spoofed packets to the node at high data rates. This can be mitigated by connecting ATN/IPS ASBRs over dedicated links with no connections to the Internet and/or when ASBR connections to the Internet are only permitted through well-managed firewalls.

ATN/IPS s-ASBRs should institute rate limits to protect low data rate aviation data links from receiving DDoS packet floods.

BGP protocol message exchanges and control message exchanges used for route optimization must be secured to ensure the integrity of the system-wide routing information base.

This document does not include any new specific requirements for mitigation of DDoS.

11. Acknowledgements

This work is aligned with the FAA as per the SE2025 contract number DTFAWA-15-D-00030.

This work is aligned with the NASA Safe Autonomous Systems Operation (SASO) program under NASA contract number NNA16BD84C.

This work is aligned with the Boeing Information Technology (BIT) MobileNet program.

The following individuals contributed insights that have improved the document: Erik Kline, Hubert Kuenig, Tony Li, Alexandre Petrescu, Pascal Thubert, Tony Whyman.
12. References

12.1. Normative References


12.2. Informative References


Appendix A. BGP Convergence Considerations

Experimental evidence has shown that BGP convergence time required for when an MNP is asserted at a new location or withdrawn from an old location can be several hundred milliseconds even under optimal AS peering arrangements. This means that packets in flight destined to an MNP route that has recently been changed can be (mis)delivered to an old s-ASBR after a Client has moved to a new s-ASBR.

To address this issue, the old s-ASBR can maintain temporary state for a "departed" Client that includes a SPAN address for the new s-ASBR. The SPAN address never changes since ASBRs are fixed infrastructure elements that never move. Hence, packets arriving at the old s-ASBR can be forwarded to the new s-ASBR while the BGP routing system is still undergoing reconvergence. Therefore, as long as the Client associates with the new s-ASBR before it departs from the old s-ASBR (while informing the old s-ASBR of its new location) packets in flight during the BGP reconvergence window are accommodated without loss.
Appendix B. Change Log

<< RFC Editor - remove prior to publication >>

Changes from -01 to -02:

- introduced the SPAN and the concept of Internetwork partitioning
- new terms "ANET" (for (Radio) Access Network) and "INET" (for Internetworking underlay)
- new appendix on BGP convergence considerations

Changes from -00 to -01:

- incorporated clarifications due to list comments and questions.
- new section 7 on Stub AS Mobile Routing Services
- updated references, and included new reference for MIPv6 and LISP

Status as of 08/30/2018:

- 'draft-templin-atn-bgp' becomes 'draft-ietf-rtgwg-atn-bgp'

Authors' Addresses

Fred L. Templin (editor)
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: fltemplin@acm.org

Greg Saccone
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: gregory.t.saccone@boeing.com
Gaurav Dawra  
LinkedIn  
USA  
Email: gdawra.ietf@gmail.com

Acee Lindem  
Cisco Systems, Inc.  
USA  
Email: acee@cisco.com

Victor Moreno  
Cisco Systems, Inc.  
USA  
Email: vimoreno@cisco.com
Scalable De-Aggregation for Overlays Using the Border Gateway Protocol (BGP)
draft-templin-rtgwg-scalable-bgp-01.txt

Abstract

The Border Gateway Protocol (BGP) has well-known limitations in terms of the numbers of routes that can be carried and stability of the routing system. This is especially true when mobile nodes frequently change their network attachment points, which in the past has resulted in excessive announcements and withdrawals of de-aggregated prefixes. This document discusses a means of accommodating scalable de-aggregation of IPv6 prefixes for overlay networks using BGP.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 2, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

The Border Gateway Protocol (BGP) [RFC4271] has well-known limitations in terms of the numbers of routes that can be carried and the stability of the routing system. This is especially true for routing systems that include mobile nodes that frequently change their network attachment points, which in the past have resulted in excessive announcements and withdrawals of de-aggregated prefixes. This document discusses a means of accommodating scalable de-aggregation of IPv6 prefixes [RFC8200] for overlay networks using BGP.

2. Overview and Analysis

As discussed in [I-D.ietf-rtgwg-atn-bgp] and [I-D.templin-intarea-6706bis], the method for accommodating de-aggregation is to institute an overlay network instance of BGP that is separate and independent from the global Internet BGP routing system. The overlay is presented to the global Internet as a small number of aggregated IPv6 prefixes (also known as Mobility Service Prefixes (MSPs)) that never change. In this way, the Internet BGP routing system sees only stable aggregated MSPs (e.g., 2001:db8::/32).
and is completely unaware of any de-aggregation or mobility-related churn that may be occurring within the overlay.

The overlay is operated by an Overlay Service Provider (OSP), and consists of a core Autonomous System (AS) with core AS Border Routers (c-ASBRs) that connect to stub ASes with stub ASBRs (s-ASBRs) in a hub-and-spokes fashion. Mobile nodes associate with nearby (i.e., regional) stub ASes for extended timeframes, and change to new stub ASes only after movements of significant topological or geographical distance. Mobility-related changes between stub ASes are therefore normally infrequent.

The s-ASBRs use eBGP to announce de-aggregated Mobile Network Prefixes (MNPs) of mobile nodes (e.g., 2001:db8:1:2::/64, etc.) to their neighboring c-ASBRs, but do not announce fine-grained mobility events such as a mobile node moving to a new network attachment point. Instead, mobile nodes coordinate with stub ASes using mobility protocols such as MIPv6, LISP, AERO, etc. and stub ASes accommodate these localized mobility events without disturbing the c-ASBRs.

The c-ASBRs originate "default" to their neighboring s-ASBRs but do not announce any MNP routes. In this way, MNP announcements and withdrawals are unidirectional from s-ASBRs to c-ASBRs only, thereby suppressing BGP updates on the reverse path. The c-ASBRs in turn use iBGP to maintain a consistent view of the full topology. BGP Route Reflectors (RRs) [RFC4456] can also be used to support increased c-ASBR scaling.

Each c-ASBR should be able to carry at least as many routes as a typical core router in the global public Internet BGP routing system. Since the number of active routes in the Internet is rapidly approaching 1 million (1M), viable c-ASBRs must be capable of carrying at least 1M MNP routes (this has been proven even for BGP running on lightweight virtual machines). The method for increasing scaling therefore is to divide the MSP into longer sub-MSPs, and to assign a different set of c-ASBRs for each sub-MSP.

For example, the MSP 2001:db8::/32 could be sub-divided into sub-MSPs such as 2001:db8:0010::/44, 2001:db8:0020::/44, 2001:db8:0030::/44, etc. with each sub-MSP assigned to a different set of c-ASBRs. Each s-ASBR peers with at least one member of each c-ASBR set and uses route filters such that BGP updates are only sent to the c-ASBR(s) that aggregate the specific sub-MSP. Then, assuming 1 thousand (1K) or more sub-MSPs (each with its own set of c-ASBRs) the entire BGP overlay routing system should be able to service 1 billion (1B) MNPs or more.
3. Opportunities and Limitations

Since a lightweight virtual machine (e.g., a linux image running quagga in the cloud) can service up to 1M MNPs using BGP, it is likely that dedicated high-performance IPv6 router hardware could support even more. With such dedicated high-performance hardware, the number of MNPs could be increased further.

The deployed numbers of s-ASBRs even for very large overlays should not exceed a c-ASBR’s capacity for BGP peering sessions. For example, c-ASBRs should be capable of servicing 1K or more BGP peering sessions, with the upper bound limited by keepalive and update control messaging overhead. Conversely, s-ASBRs should be capable of supporting even more sessions since they only receive keepalives and only send updates for mobile nodes within their local stub ASes.

Mobile nodes should refrain from moving rapidly between stub ASes for no good reason, since the objective is only to reduce routing stretch due to movement of significant distances. OSPs could employ disincentives such as surcharge penalties for gratuitous mobility, but intentional abuse would also yield little reward since only the bad actor (i.e., and not others) would be subject to MNP instability.

Packets sent between mobile nodes that associate with different stub ASes would initially need to be forwarded through the core AS, which presents a forwarding bottleneck. For this reason, a route optimization function is needed to reduce congestion in the core. Since c-ASBRs should be commercial off-the-shelf (COTS) dedicated high-performance IPv6 routers, however, they should not be required to participate directly in any out-of-band route optimization signaling. Instead, route optimization should be coordinated by stub AS network elements and/or the mobile nodes themselves.

4. Use Cases

Use cases include Unmanned Air Systems (UAS) in controlled and uncontrolled airspaces, Intelligent Transportation Systems (ITS) in urban air/ground mobility environments, aviation networks, enterprise mobile device users, and cellular network users. Any other use cases in which an OSP services large numbers of mobile nodes are also in scope.

5. Implementation Status

The arrangement of stub and core ASes described in this document has been implemented using standards-compliant linux operating systems and BGP routing protocol implementations (i.e., quagga). No new code
was included, and all requirements were satisfied through standard configuration options.

6. IANA Considerations

This document does not introduce any IANA considerations.

7. Security Considerations

Security considerations are discussed in the references.

8. Acknowledgements

This work is aligned with the FAA as per the SE2025 contract number DTFAWA-15-D-00030.

This work is aligned with the NASA Safe Autonomous Systems Operation (SASO) program under NASA contract number NNA16BD84C.

This work is aligned with the Boeing Information Technology (BIT) MobileNet program.

9. References

9.1. Normative References


9.2. Informative References

Appendix A. Change Log

Changes from -00 to -01:

- added Route Reflectors
- introduced term "Overlay Service Provider (OSP)"
- removed estimate of number of routes for high-performance routers
- revised text on route optimization
- added use case and implementation sections

Status as of 01/23/2018:

- -00 draft published

Authors’ Addresses

Fred L. Templin (editor)
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: fltemplin@acm.org

Greg Saccone
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: gregory.t.saccone@boeing.com