DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 7, 2015

M. Liebsch
NEC
S. Matsushima
Softbank Telecom
S. Gundavelli
Cisco
D. Moses
Intel Corporation

May 6, 2015

Protocol for Forwarding Policy Configuration (FPC) in DMM
draft-ietf-dmm-fpc-cpdp-00.txt

Abstract

The specification as per this document supports the separation of the Control-Plane for mobility- and session management from the actual Data-Plane. The protocol semantics abstract from the actual details for the configuration of Data-Plane nodes and apply between a Client function, which is used by an application of the mobility Control-Plane, and an Agent function, which is associated with the configuration of Data-Plane nodes according to the policies issued by the mobility Control-Plane. The scope of the policies comprises forwarding rules and treatment of packets in terms of encapsulation, IP address re-writing and QoS. Additional protocol semantics are described to support the maintenance of the Data-Plane path.

Table of Contents

1. Introduction

One objective of the Distributed Mobility Management (DMM) WG is the separation of the mobility management Control- and Data-Plane to enable flexible deployment, such as decentralized provisioning of Data-Plane nodes (DPN). Data-Plane nodes can be configured to function as anchor for a registered Mobile Node's (MN) traffic, others can be configured to function as Mobile Access Gateway (MAG) as per the Proxy Mobile IPv6 protocol [RFC5213] or a Foreign Agent

(FA) as per the Mobile IPv4 protocol [RFC3344]. Requirements for DMM have been described in [RFC7333], whereas best current practices for DMM are documented in [RFC7429].

The Data-Plane must provide a set of functions to the Mobility Control-Plane, such as support for encapsulation, IP address re-writing, QoS differentiation and traffic shaping. In addition, the configuration of forwarding rules must be provided. These requirements are met by various transport network components, such as IP switches and routers, though configuration semantics differs between them.

Forwarding Policy Configuration (FPC) as per this document enables the configuration of any Data-Plane node and type by the abstraction of configuration details and the use of common configuration semantics. The protocol using the FPC semantics is deployed between a Client function, which is associated with the Mobility Management Control-Plane, and an Agent function. The Agent function enforces the Data-Plane configuration and can be present on a transport network controller or co-located with a Data-Plane node. The Agent applies the generalized configuration semantics to configuration, which is specific to the Data-Plane node and type. The Mobility Control-Plane can select one or multiple DPNs which suit the MN's mobility management without the need to handle each node's routing- or switching tables and local interface configurations for potentially many routers serving the Data-Plane, but enforce the policies for traffic treatment and forwarding through the FPC Client and the FPC Agent functions.

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].

3. Model for Policy-based DMM Network Control

3.1. Reference Architecture for DMM Forwarding Policy Configuration

The DMM Forwarding Policy Configuration (FPC) protocol enables DMM use cases in deployments with separated Control-/Data-Plane and is used by applications of the Mobility Control-Plane to enforce rules for forwarding and traffic treatment in the Data-Plane. Figure 1 depicts an exemplary use case where downlink traffic from a Correspondent Node (CN) towards a Mobile Node (MN) traverses multiple DPNs, each applying policies as per the Control-Plane's request. Policies in the one or multiple DPNs can result in traffic steering according to a host-route, packet scheduling and marking according to

a subscriber's QoS profile, or forwarding rules (e.g. encapsulation within GRE or GTP-U tunnel).

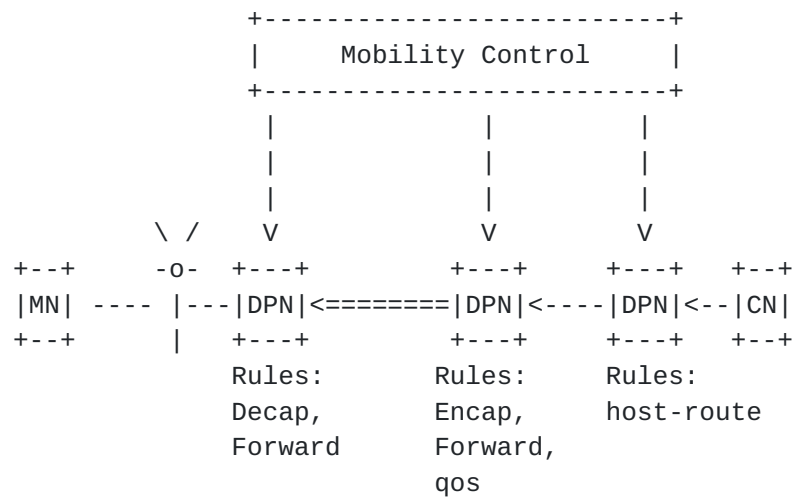

Figure 1: Exemplary illustration of a use case for DMM traffic steering and policy enforcement at Data Plane Nodes (DPN)

Mobility Control-Plane functions have the following roles in common:

- Tracking an MN's location
- Accept requests to set up and maintain mobility-related Data-Plane path between DPNs, taking QoS attributes into account. Such requests can be issued through mobility protocols, such as Proxy Mobile IPv6, and the associated operation with remote Mobility Control-Plane functions.
- Become aware of different DPNs that provide the required Data-plane functions to the Mobility Control-Plane and can be used for mobility traffic forwarding and treatment
- Monitor the DPNs' operation and handle exceptions, e.g. the detection of a partial DPN failure and the diversion of traffic through a different DPN
- Maintain consistency between multiple DPNs which enforce policy rules for an MN

Mobility Data-Plane functions have the following roles in common:

- Forward and treat traffic according to the policies and directives sent by the Mobility Control-Plane

- Provide status (e.g. load, health, statistics and traffic volume) information on request

- Participate in the process for topology acquisition, e.g. by exposing relevant topological and capability information, such as support for QoS differentiation and supported encapsulation protocols

The protocol for DMM FPC applies to the interface between an FPC Client function and an FPC Agent function, as depicted in Figure 2. The FPC Client function is associated with an application function of the mobility management Control-Plane, e.g. a Local Mobility Anchor Control-Plane function as per the Proxy Mobile IPv6 protocol. The FPC Agent function processes the FPC protocol semantics and translates them into configuration commands as per the DPN's technology. In one example, an FPC Agent can be co-located with a Transport Network Controller, which enforces forwarding rules on a set of SDN switches. In another example, the Agent can be co-located with a single router to directly interact with interface management and the router's RIB Manager. The mapping of the common FPC semantics and policy description as per this specification to the configuration commands of a particular DPN is specific to the DPN's technology and the Agent's implementation.

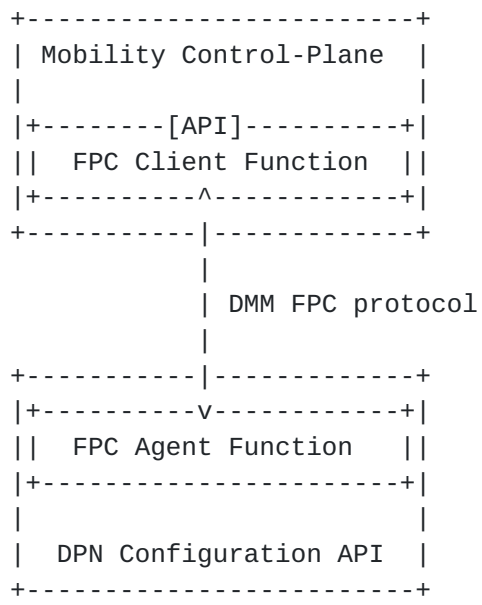

Figure 2: Illustration of the functional reference architecture for DMM Forwarding Policy Configuration (FPC)

3.2. Generalized Rules on the Client-Agent-Interface

To abstract configuration details of an IP switch or IP router on the FPC protocol interface, this specification adopts the model of logical gates (Ports) to bind certain properties, such as a QoS policy. Additional properties can be bound to the same logical Port, e.g. encapsulation of packets, being directed to that logical Port, in a GRE tunnel. The remote tunnel endpoint is configured as part of the property bound to that logical Port. All traffic, which has a forwarding rule in common and should be forwarded according to the properties bound to a particular Port, can be referred to that Port by configuration of a forwarding rule. Multiple IP flows or even aggregated traffic being destined to a given IP prefix can be directed to that logical Port and experiences the same treatment according to the configured properties and forwarding characteristics. Aggregated or per-Host/per-Flow traffic can be identified by a longest prefix match or a Traffic Selector respectively.

Figure 3 illustrates the generic policy configuration model as used between an FPC Client function and an FPC Agent function.

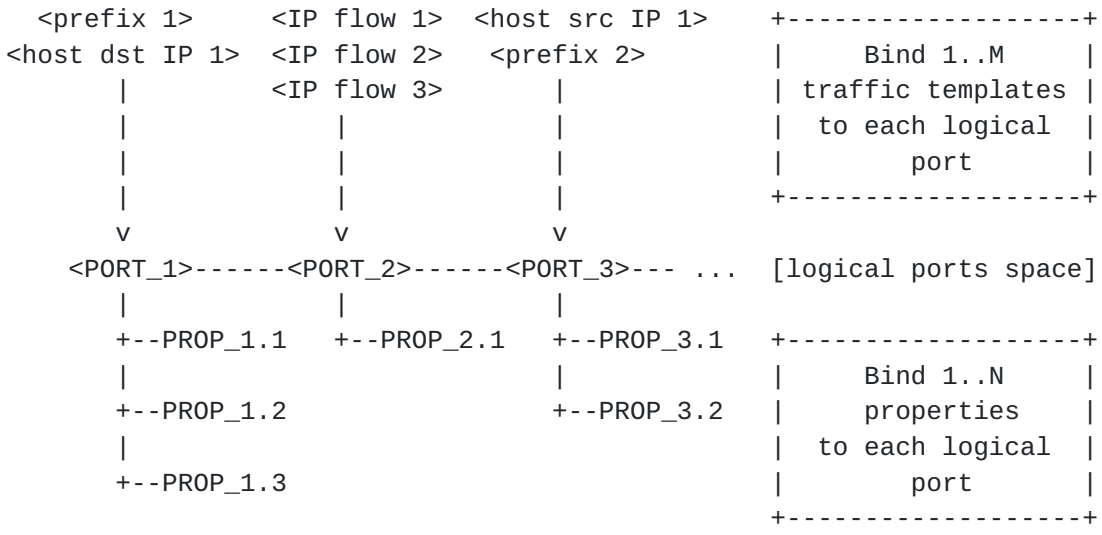

Figure 3: Illustration of generalized rules

3.3. Role of the DMM FPC Client Function

The DMM FPC Client function includes the following tasks:

- Per mobility management transaction or relevant event, build one or multiple Control messages/attributes to control policies on one or multiple DPA(s) according to the application's directives

- Treat a DPN's policy rules (encapsulation, address re-write, QoS, traffic monitoring) on the basis of properties being bound to logical ports (similar to the bearer concept in cellular networks)

- Build, modify or delete logical ports as needed

- Bind associated policy rules as one or multiple properties to a logical port

- Treat forwarding rules (e.g. per-IP flow, per-MN, per-IP, per-prefix) on the basis of logical ports

- Send each generated message to the DMM FPC Agent associated with the identified DPN

- Keep record of the policy rules/port information and the associated DPN and FPC Agent Function

- Process received Response, Notification and Query messages issued by a DMM FPC Agent Function and notify the application

3.4. Role of the DMM FPC Agent Function

The DMM FPC Agent function includes the following tasks:

- Process the received Control messages issued by a DMM FPC Client Function

- Unambiguously match each logical port with an associated physical port or interface at the identified DPN

- Apply the received properties to local configuration (e.g. encapsulation, NA(P)T, traffic prioritization and scheduling) on the identified DPN according to the DPN's technology

- Monitor scheduled events (e.g. failure or missing rule) and issue an associated message to the FPC Client Function (NOTIFICATION, QUERY)

4. Protocol Messages and Semantics

4.1. Protocol Messages

Message	Description
Messages issued by the FPC Client	
PRT_ADD	Add a logical port
PRT_DEL	Delete an existing logical port
PROP_ADD	Add a property to a logical port
PROP_MOD	Modify a property of a logical port
PROP_DEL	Remove and delete a property from a logical port
RULE_ADD	Add forwarding rule by binding traffic descriptor to a logical port
RULE_MOD	Modify existing forwarding rule by changing the traffic descriptor bound to a logical port
RULE_DEL	Delete a forwarding rule
EVENT_REG	Register an event at an Agent, which is to be monitored by the Agent and to be reported
PROBE	Probe the status of a registered event
Messages issued by the FPC Agent	
NOTIFY	Notify the Client about the status of a monitored attribute at any event kind (periodic / event trigger / probed)
QUERY	Query the Client about missing rules/states

Figure 4: Protocol Messages

4.2. Protocol Attributes

Protocol messages as per Section 4.1 carry attributes to identify an FPC Client- or Agent function, as well as a DPN, logical ports and configuration data. Furthermore, attributes are carried to manage logical ports and describe properties associated with a logical port, as well as to describe per-host-, aggregate or IP flow traffic and refer to a logical port as forwarding information.

This document specifies attributes from the following categories:

- Identifier attributes
- Properties
- Property-specific attributes
- Traffic descriptors

Note on the list of attributes: The list of attributes is not yet complete.

Note on Format Clarification: Meant to provide a first idea on the format and number space and indicates length (bit) and semantics of key information fields.

Attribute	Format Clarification	Description
	Identifiers	
PRT_ID	[16,PTR_ID]	Identifies a logical Port
PRT_PROP_ID	[16,PRT_ID] [8,PROP_ID]	Identifies a logical Port and one of its properties
CLI_ID	[8, Carrier ID] [8, Network ID] [16, Client ID]	Identifies an FPC Client function
AGT_ID	[8, Carrier ID] [8, Network ID] [16, Agent ID]	Identifies an FPC Agent function
DPN_ID	[8, Carrier ID] [8, Network ID] [16, DPN ID]	Identifies a Data Plane Node (DPN)
EVENT_ID	[16, Event ID]	Identifies a registered event

Figure 5: Protocol Attributes: Identifiers

Attribute	Format Clarification	Description
Properties		
PROP_TUN	[type][src][dst]	Property Encapsulation, indicates type GRE, IP, GTP
PROP_REWR	TBD	Property NAT
PROP_QOS	TBD	Property QoS
PROP_GW	[ip address next hop]	Property Next Hop

Figure 6: Protocol Attributes: Properties

Attribute	Format Clarification	Description
Property-specific		
IPIP_CONF		IP-encapsulation configuration attribute
GRE_CONF	[prototype][seq-#] [key]..	GRE_encapsulation configuration attribute
GTP_CONF	[TEID_local] [TEID_remote] [seq-#]..	GTP-U encapsulation configuration attribute

Figure 7: Protocol Attributes: Property-specific

4.3. Protocol Operation

The following list comprises a more detailed description of each message's semantic.

- PRT_ADD - Issued by a Client to add a new logical port at an Agent, to which traffic can be directed. An Agent receiving the PRT_ADD message should identify the new logical port according to the included port identifier (PRT_ID). In case the DPN holds already a registration for a logical port with the same identifier, the Agent should throw an error message to the Client. Otherwise the Agent should add a new logical port into its conceptual data structures using the port identifier as key.

- PRT_DEL - Used by a Client to delete an existing logical port. An Agent receiving such message should delete all properties associated with the identified port.

- PROP_ADD - Used by the Client to add a new property to an existing logical port. The property is unambiguously identified through a property identifier (PRT_PROP_ID). All traffic, which is directed to this logical port, experiences the existing and newly added property.

- PROP_MOD - Used by a Client to modify an existing property. For example, a tunnel property can be changed to direct traffic to a different tunnel endpoint in case of an MN's handover

- PROP_DEL - Used by a Client to delete one or multiple properties, each being identified by a property identifier.

- RULE_ADD - Used by a Client to add a forwarding rule and direct traffic towards a logical port. The rule add command must unambiguously identify aggregated traffic (longest prefix), per host IP traffic or per-flow traffic in the RULE_ADD command and bind the identified traffic to a logical port. An Agent receiving a RULE_ADD command must add the rule to its local conceptual data structures and apply commands for local configuration to add the new forwarding rule on the DPN. Multiple forwarding rules, each identifying different traffic, can direct traffic to the same logical port. All traffic being directed to this logical port will then experience the same properties.

- RULE_MOD - Used by a Client to modify an existing forwarding rule. An Agent receiving such message should apply commands for local configuration to update the forwarding rule on the DPN.

- RULE_DEL - Used to delete an existing forwarding rule on a DPN. The Agent receiving such message should delete the rules from its local conceptual data structures and apply commands for local configuration to remove the forwarding rule on the DPN.

- EVENT_REG - Used by a Client to register an attribute, which is to be monitored, at an Agent. The EVENT_REG provides an attribute to the Agent as well as a reporting kind. The Agent should register the event and an event identifier in the local conceptual data structures. The Agent should start monitoring the registered attribute (e.g. load) and notify the Client about the status according to the registered reporting kind (periodic, event trigger, probed). In case of a periodic reporting kind, the Agent should report the status of the attribute each configured interval using a NOTIFY message. The reporting interval is provided with

the EVENT_REG message. In case of an event triggered reporting kind, the Agent should report the status of the attribute in case of a triggered event, e.g. the monitored attribute's value exceeds a given threshold. The threshold is provided with the EVENT_REG message. In case of probed reporting, the Agent receives a PROBE message and should report the status of a monitored attributes to the Client by means of a NOTIFY message.

- PROBE - Used by a Client to retrieve information about a previously registered event. The PROBE message should identify one or more events by means of including the associated event identifier. An Agent receiving a PROBE message should send the requested information for each event in a single or multiple NOTIFY messages.

- NOTIFY - Used by an Agent to report the status of an event to a Client.

- QUERY - Used by an Agent to request an update of logical port properties via a Client.

Figure 8 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.

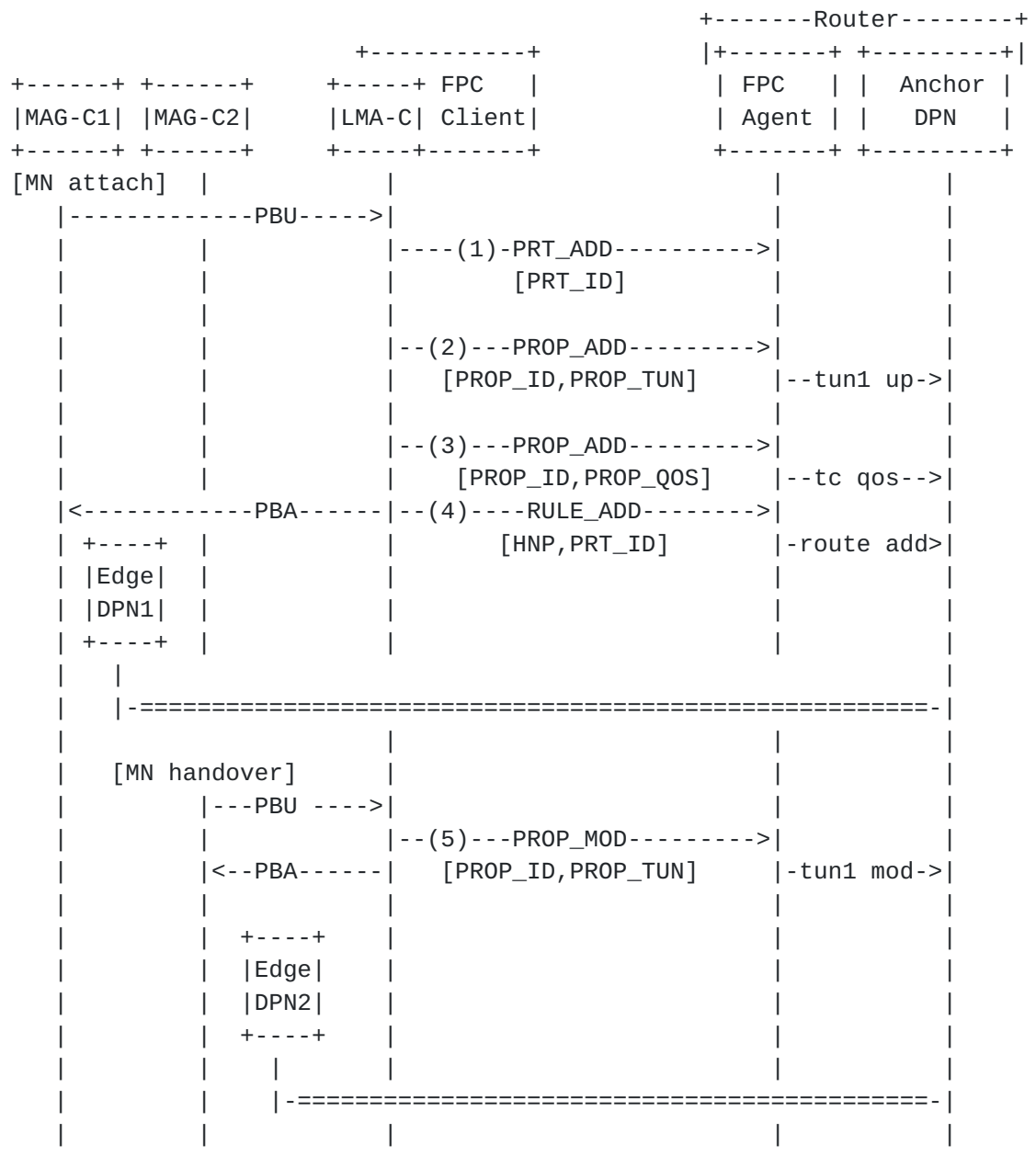

Figure 8: Exemplary Message Sequence (focus on FPC reference point)

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

Subsequently, the LMA-C adds properties to the new logical port. One property is added (2) to specify the forwarding tunnel type and endpoints (Anchor DPN, Edge DPN1). Another property is added (3) to specify the QoS differentiation, which the MN's traffic should experience. At reception of the properties, the FPC Agent calls local router commands to enforce the tunnel configuration (tun1) as well as the traffic control (tc) for QoS differentiation. After configuration of port properties have been completed, the LMA can configure the enforcement of the MN's traffic by adding a rule (RULE_ADD) to forward traffic destined to the MN's HNP to the new logical port (4). At the reception of the forwarding rule, the Agent applies a new route to forward all traffic destined to the MN's HNP to the configured tunnel interface (tun1).

During handover, the LMA-C receives an updating PBU from the handover target MAG-C2. The PBU refers to a new Data-Plane node (Edge DPN2) to represent the new tunnel endpoint. The LMA-C sends a PROP_MOD message (5) to the Agent to modify the existing tunnel property of the existing logical port and to update the tunnel endpoint from Edge DPN1 to Edge DPN2. At reception of the PROP_MOD message, the Agent applies local configuration commands to modify the tunnel.

To reduce the number of protocol handshakes between the LMA-C and the DPN, the LMA-C can append property (PROP_TUN, PROP_QOS) and rules (prefix info HNP) attributes to the PRT_ADD message, as illustrated in Figure 9

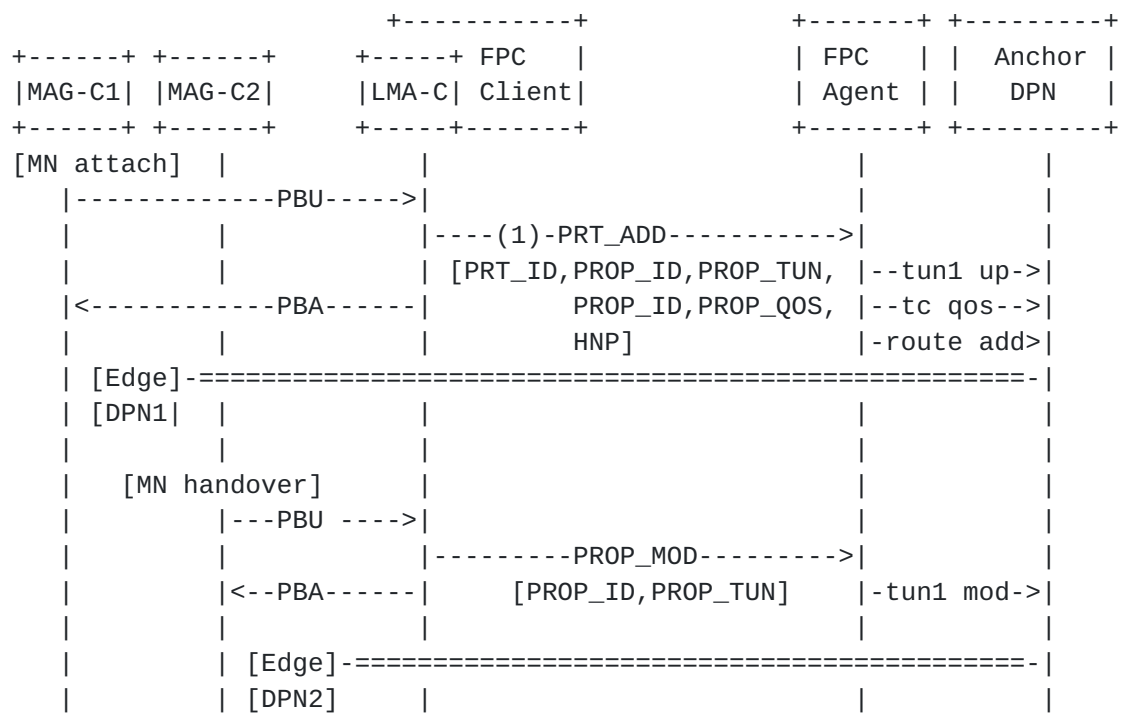

Figure 9: Example: Sequence for Message Aggregation (focus on FPC reference point)

5. Conceptual Data Structures

An FPC Client must keep record about the logical ports, each port's properties as well as configured rules as per the Mobility Control-Plane function's request. Such information must be maintained for each Agent, with which the Client communicates. In case the Mobility Control-Plane function identifies a particular DPN at which the policies should be enforced, the Client must associate the DPN identifier with the logical port configuration.

According to the FPC Agent's role, the Agent translates the generalized model for policy configuration and forwarding rules into semantics and commands for local configuration, which is specific to a DPN. Keeping a local record of DPN configuration attributes/values is implementation specific and out of scope of this document.

Description of detailed data structures and information to be recorded and maintained by an FPC Client and an FPC Agent are TBD and will be added to a revision of this initial document.

6. Security Considerations

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

7. IANA Considerations

This document provides an information model for DMM Forwarding Policy Configuration. Detailed protocol specifications for DMM Forwarding Policy Configuration will follow the information model as per this document and can be based on, for example, ReST-like or binary protocol formats. Such protocol-specific details will be described in separate documents and may require IANA actions.

8. 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.

9. References

9.1. Normative References

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

[RFC7333] Chan, H., Liu, D., Seite, P., Yokota, H., and J. Korhonen, "Requirements for Distributed Mobility Management", RFC 7333, August 2014.

[RFC7429] Liu, D., Zuniga, JC., Seite, P., Chan, H., and CJ. Bernardos, "Distributed Mobility Management: Current Practices and Gap Analysis", RFC 7429, January 2015.

9.2. Informative References

[RFC3344] Perkins, C., "IP Mobility Support for IPv4", RFC 3344, August 2002.

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

Appendix A. YANG Data Model for the FPC Protocol

This appendix provides (so far experimental) formating of some FPC protocol components adopting YANG data modeling. The current FPC information model as per this initial draft version will experience extensions, as it is not yet complete, and may experience changes that need to be reflected in the data model. Whether a detailed data model will be included in this document or solely an information model will be adopted by this document and a detailed data model will be part of a separate document is currently being discussed.

```
module ietf-dmm-fpcp {
    namespace "urn:ietf:params:xml:ns:yang:dmm-fpcp";
    prefix fpcp;

    import ietf-inet-types { prefix inet; }

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

    revision 2015-03-09 {}

    typedef fpcp-port-id {
        description "PRT_ID";
        type uint16;
    }

    typedef fpcp-property-id {
        description "PROP_ID";
        type uint8;
    }

    identity tunnel-type {
        description
        "Base identity from which specific use of
         tunnels are derived.";
    }

    identity fpcp-tunnel-type {
        base "tunnel-type";
        description
        "Base identity from which specific tunnel
         types in FPCP uses are derived.";
    }

    identity ip-in-ip {
```

```
            base "fpcp-tunnel-type";
            description "IP-in-IP tunnel";
        }

        identity gtp {
            base "fpcp-tunnel-type";
            description "GTP-U tunnel";
        }

        identity gre {
            base "fpcp-tunnel-type";
            description "GRE tunnel";
        }

        identity ip-protocol {
            description
            "Base identity from which specific
             IP protocol types are derived.";
        }

        identity qos-type {
            description
            "Base identity from which specific
             uses of QoS types are derived.";
        }

        identity fpcp-qos-type {
            base "qos-type";
            description
            "Base identity from which specific
             QoS types in FPCP uses are derived.";
        }

        identity fpcp-qos-type-high {
            base "fpcp-qos-type";
            description
            "An example FPCP QoS Type for high quality class.
             FPCP supported QoS classes are TBD.";
        }

        identity fpcp-qos-type-middle {
            base "fpcp-qos-type";
            description
            "An example FPCP QoS Type for middle quality class.
             FPCP supported QoS classes are TBD.";
        }

        identity fpcp-qos-type-low {
```

```
            base "fpcp-qos-type";
            description
            "An example FPCP QoS Type for low quality class.
             FPCP supported QoS classes are TBD.";
        }

        grouping fpcp-client {
            description "CLI_ID to identify FPCP Client";
            leaf carrier-id {
                type uint8;
            }
            leaf network-id {
                type uint8;
            }
            leaf client-id {
                type uint16;
                mandatory true;
            }
        }

        grouping fpcp-agent {
            description "AGT_ID to identify FPCP Agent";
            leaf carrier-id {
                type uint8;
            }
            leaf network-id {
                type uint8;
            }
            leaf agent-id {
                type uint16;
                mandatory true;
            }
        }

        grouping dpn {
            description "DPN_ID to identify Data-Plane Node";
            leaf carrier-id {
                type uint8;
            }
            leaf network-id {
                type uint8;
            }
            leaf dpn-id {
                type uint16;
                mandatory true;
            }
        }
```

```
    grouping port-property-id {
        description "PRT_PROP_ID";
        leaf port-id {
            mandatory true;
            type fpcp-port-id;
        }
        leaf property-id {
            type fpcp-property-id;
            mandatory true;
        }
    }

    grouping tunnel-endpoints {
        description
        "PROP_TUN property as a set of tunnel endpoints";
        leaf tunnel-type {
            type identityref {
                base "fpcp-tunnel-type";
            }
        }
        leaf remote-address {
            type inet:ip-address;
        }
        leaf local-address {
            type inet:ip-address;
        }
    }

    grouping gtp-attributes {
        description
        "GTP_CONF as GTP tunnel specific attributes";
        leaf remote-teid {
            type uint32;
        }
        leaf local-teid {
            type uint32;
        }
    }

    grouping gre-attributes {
        description
        "GRE_CONF as GRE tunnel specific attribute";
        leaf key {
            type uint32;
        }
    }

    grouping fpcp-identifier-attributes {
```

```
            description
            "Identifiers of protocol attributes";
            leaf port-id {
                type fpcp-port-id;
            }
            container client {
                uses fpcp-client;
            }
            container agent {
                uses fpcp-agent;
            }
            list nodes {
                key dpn-id;
                uses dpn;
            }
        }

        grouping fpcp-traffic-descriptor {
            description
            "Traffic descriptor group collects parameters to
             identify target traffic flow and apply QoS policy";
            leaf destination-ip {
                type inet:ip-prefix;
            }
            leaf source-ip {
                type inet:ip-prefix;
            }
            leaf protocol {
                type identityref {
                    base "ip-protocol";
                }
            }
            leaf destination-port {
                type inet:port-number;
            }
            leaf source-port {
                type inet:port-number;
            }
            leaf qos {
                type identityref {
                    base "fpcp-qos-type";
                }
            }
        }

        grouping fpcp-port-properties {
            description
            "A set of port property attributes";
```

```
            leaf property-id {
                type fpcp-property-id;
            }
            list next-hops {
                container endpoints {
                    uses tunnel-endpoints;
                }
                choice tunnel {
                    case gtp-u {
                        when "tunnel-type = 'gtp'";
                        uses gtp-attributes;
                    }
                    case gre {
                        when "tunnel-type = 'gre'";
                        uses gre-attributes;
                    }
                }
            }
        }


        // Port Entries

        container port-entries {
            description
            "This container binds set of traffic-descriptor and
             port properties to a port and lists them as a port entry.";
            list port-entry {
                key port-id;
                container identifier {
                    uses fpcp-identifier-attributes;
                }
                container trafic-descriptor {
                    uses fpcp-traffic-descriptor;
                }
                list properties {
                    uses fpcp-port-properties;
                }
            }
        }

        // PRT_ADD

        rpc port_add {
            description "PRT_ADD";
            output {
                list fpcp-port-entry {
                    uses fpcp-identifier-attributes;
```

```
            }
        }
    }

    // PRT_DEL

    rpc port_delete {
        description "PRT_DEL";
        input {
            leaf deleting-port {
                type fpcp-port-id;
            }
        }
    }

    // PROP_ADD

    rpc port_property_add {
        description "PROP_ADD";
        input {
            leaf target-port {
                type fpcp-port-id;
                mandatory true;
            }
            container port-properties {
                uses fpcp-port-properties;
            }
        }
    }

    // PROP_MOD

    rpc port_property_modify {
        description "PROP_MOD";
        input {
            leaf target-port {
                type fpcp-port-id;
                mandatory true;
            }
            container port-properties {
                uses fpcp-port-properties;
            }
        }
    }

    // PROP_DEL

    rpc port_property_delete {
```

```
            description "PROP_DEL";
            input {
                container deleting-property {
                    uses port-property-id;
                }
            }
        }

        // RULE_ADD

        rpc rule_add {
            description
            "TBD for input parameters of which RULE_ADD includes
             but now just traffic-descriptor.";
            input {
                leaf target-port {
                    type fpcp-port-id;
                    mandatory true;
                }
                container port-properties {
                    uses fpcp-traffic-descriptor;
                }
            }
        }

        // RULE_MOD

        rpc rule_modify {
            description
            "TBD for input parameters of which RULE_MOD includes
             but now just traffic-descriptor.";
            input {
                leaf target-port {
                    type fpcp-port-id;
                    mandatory true;
                }
                container port-properties {
                    uses fpcp-traffic-descriptor;
                }
            }
        }

        // RULE_DEL

        rpc rule_delete {
            description
            "TBD for input parameters of which RULE_DEL includes
             but now just traffic-descriptor.";
```

```
            input {
                leaf target-port {
                    type fpcp-port-id;
                    mandatory true;
                }
                container port-properties {
                    uses fpcp-traffic-descriptor;
                }
            }
        }

        // EVENT_REG

        rpc event_register {
            description
            "TBD for registered parameters included in EVENT_REG.";
        }

        // PROBE

        rpc probe {
            description
            "TBD for retrieved parameters included in PROBE.";
        }

        // NOTIFY

        notification notify {
            description
            "TBD for which status and event are reported to client.";
        }
    }
```

Figure 10: FPC YANG Data Model

Authors' Addresses

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

Phone: +49 6221 4342146
Email: liebsch@neclab.eu

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

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

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

Email: sgundave@cisco.com

Danny Moses

Email: danny.moses@intel.com