DMM Working Group
Internet-Draft
Intended status: Standards Track
Expires: 27 March 2021

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

23 September 2020

Protocol for Forwarding Policy Configuration (FPC) in DMM
draft-ietf-dmm-fpc-cpdp-14

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 27 March 2021.

Table of Contents

1. Introduction

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

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

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

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

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

When a DPN is mapped to physical data-plane equipment, the FPC client can have complete knowledge of the DPN architecture, and use that information to perform DPN selection for specific sessions. On the other hand, when a virtual DPN is mapped to a collection of physical DPNs, the FPC client cannot select a specific physical DPN because it is hidden by the abstraction; only the FPC Agent can address the specific associated physical DPNs. Network architects have the

flexibility to determine which DPN-selection capabilities are performed by the FPC Agent (distributed) and which by the FPC client (centralized). In this way, overlay networks can be configured without disclosing detailed knowledge of the underlying hardware to the FPC client and applications.

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

2. Terminology

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

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

Domain: One or more DPNs that form a logical partition of network resources (e.g., a data-plane network under common network administration). A FPC client (e.g., a mobility management system) may utilize a single or multiple domains.

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

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

Mobility Context: A Mobility Context contains the data-plane information necessary to efficiently send and receive traffic from a mobile node. This includes policies that are created or modified during the network's operation - in most cases, on a per-flow or per session basis. A Mobility-Context represents the mobility sessions (or flows) which are active

	on a mobile node. This includes associated runtime attributes, such as tunnel endpoints, tunnel identifiers, delegated prefix(es), routing information, etc. Mobility-Contexts are associated to specific DPNs. Some pre-defined Policies may apply during mobility signaling requests. The Mobility Context supplies information about the policy settings specific to a mobile node and its flows; this information is often quite dynamic.
Mobility Session:	Traffic to/from a mobile node that is expected to survive reconnection events.
Monitor:	A reporting mechanism for a list of events that trigger notification messages from a FPC Agent to a FPC Client.
Policy:	A Policy determines the mechanisms for managing specific traffic flows or packets. Policies specify QoS, rewriting rules for packet processing, etc. A Policy consists of one or more rules. Each rule is composed of a Descriptor and Actions. The Descriptor in a rule identifies packets (e.g., traffic flows), and the Actions apply treatments to packets that match the Descriptor in the rule. Policies can apply to Domains, DPNs, Mobile Nodes, Service-Groups, or particular Flows on a Mobile Node.
Property:	An attribute-value pair for an instance of a FPC entity.
Service-Group:	A set of DPN interfaces that support a specific data-plane purpose, e.g. inbound/ outbound, roaming, subnetwork with common specific configuration, etc.
Template:	A recipe for instantiating FPC entities. Template definitions are accessible (by name or by a key) in an indexed set. A Template is used to create specific instances (e.g., specific policies) by assigning appropriate values into the Template definition via Attribute Expression.

Template Configuration	The process by which a Template is referenced (by name or by key) and Attribute Expressions are created that change the value, default value or static nature of the Attribute, if permitted. If the Template is Extensible, new attributes MAY be added.
Tenant:	An operational entity that manages mobility management systems or applications which require data-plane functions. A Tenant defines a global namespace for all entities owned by the Tenant enabling its entities to be used by multiple FPC Clients across multiple FPC Agents.
Topology:	The DPNs and the links between them. For example, access nodes may be assigned to a Service-Group which peers to a Service-Group of anchor nodes.

3. FPC Design Objectives and Deployment

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

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

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

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

A FPC Agent configures and manages its DPN(s) according to forwarding policies requested and Attributes provided by the FPC Client. Configuration commands used by the FPC agent to configure its DPN node(s) may be specific to the DPN implementation; consequently the

method by which the FPC Agent carries out the specific configuration for its DPN(s) is out of scope for this document. Along with the data models, the FPC Client (on behalf of control-plane and applications) requests that the Agent configures Policies prior to the time when the DPNs start forwarding data for their mobility sessions.

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

This document does not specify a protocol for the FPC interface; it is out of scope.

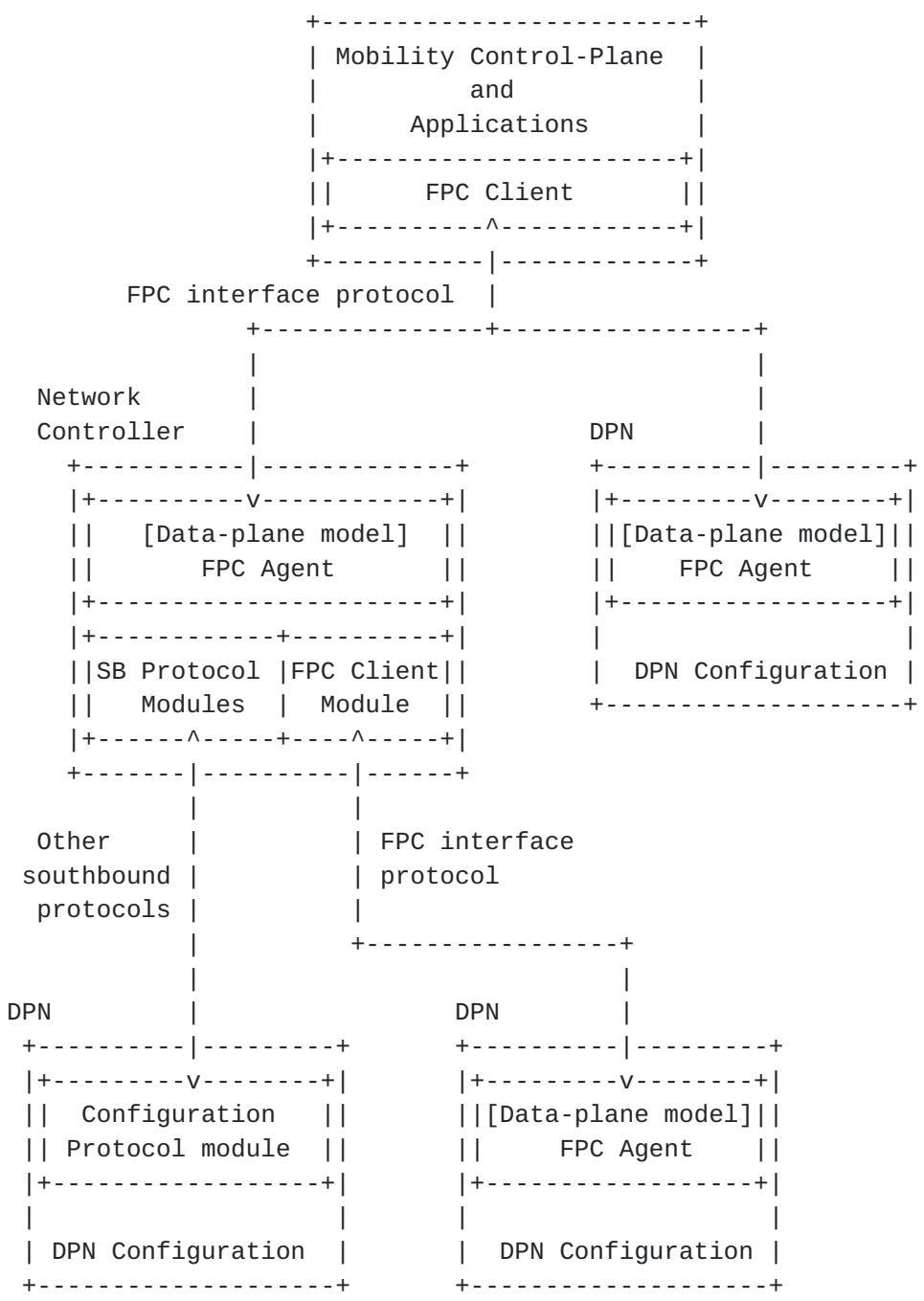
Mobility Control-Plane and Applications
FPC Client
FPC interface protocol
Network Controller
[Data-plane model]
FPC Agent
SB Protocol Modules
FPC Client Module
DPN
[Data-plane model]
FPC Agent
DPN Configuration
Other southbound protocols
FPC interface protocol
DPN
Configuration Protocol module
DPN Configuration
DPN
[Data-plane model]
FPC Agent
DPN Configuration

Figure 1: Reference Forwarding Policy Configuration (FPC) Architecture

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

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

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

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

4. FPC Mobility Information Model

The FPC information model includes the following components:

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

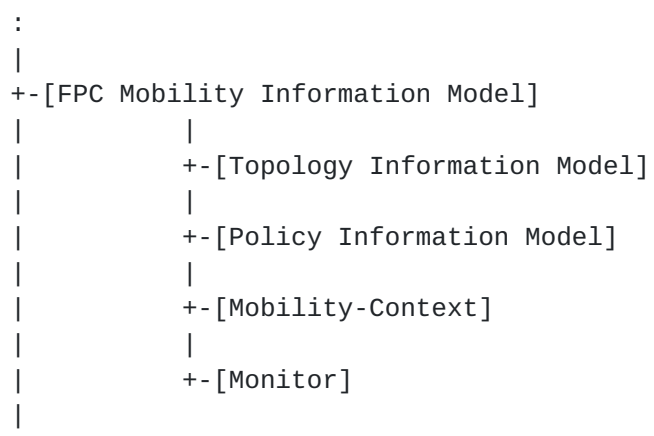

Figure 2: FPC Information Model structure

4.1. Model Notation and Conventions

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

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

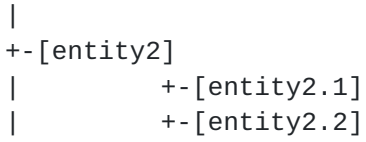

Figure 3: Model Notation - An Example

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

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

Set: A nonempty collection of entities without duplications

Name: A human-readable string

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

U-Key: A key unique across all Tenants. U-Key spaces typically

involve the use of registries or language specific mechanisms that guarantee universal uniqueness of values.

G-Key: A key unique within a Tenant

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

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

The following example shows 3 entities:

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

Figure 4

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

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

EntityA definition references an entityB model element.

EntityB model elements are indexed by entityB-Key.

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

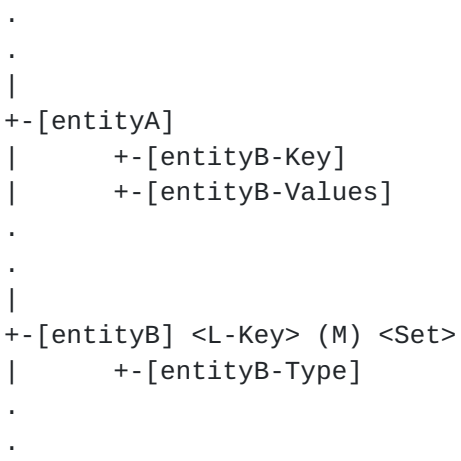

Figure 5: Indexed sets of entities

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

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

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

4.2. Templates and Attributes

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

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

Figure 6: Template entities

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

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

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

Version: Provides a version tag for the Template.

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

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

4.3. Attribute-Expressions

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

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

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

statically configured value.

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

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

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

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

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

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

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

4.4. Attribute Value Types

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

4.5. Namespace and Format

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

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

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

4.6. Configuring Attribute Values

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

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

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

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

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

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

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

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

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

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

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

Entity blocks describe the format of the policy configurations.

4.7. Entity Configuration Blocks

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

assign new values if the Attribute is not static

make attributes static if they were not

make an attribute mandatory

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

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

Figure 7: Entity Configuration Block

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

Descriptor-Configuration

Action-Configuration

Rule-Configuration

Interface-Configuration

Service-Group-Configuration

Domain-Policy-Configuration

DPN-Policy-Configuration

Policy-Configuration

MN-Policy-Configuration

Service-Data-Flow-Policy-Configuration

4.8. Information Model Checkpoint

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

Checkpoints are defined for the following information model components:

Service-Group

DPN Information Model

Domain Information Model

Policy Information Model

4.9. Information Model Components

4.9.1. Topology Information Model

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

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

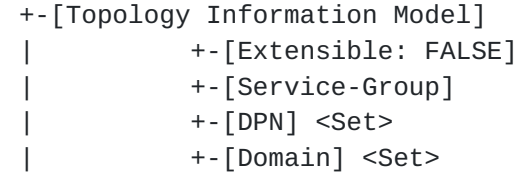

Figure 8: Topology Structure

4.9.2. Service-Group

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

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

Figure 9: Service Group

Each Service-Group element contains the following information:

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

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

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

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

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

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

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

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

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

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

4.9.3. Domain Information Model

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

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

Figure 10: Domain Information Model

Each Domain entry contains the following information:

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

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

4.9.4. DPN Information Model

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

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

Figure 11: DPN Information Model

Each DPN entry contains the following information:

DPN-Key: A unique Identifier of the DPN.

DPN-Name: A human-readable display string.

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

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

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

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

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

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

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

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

4.9.5. Policy Information Model

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

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

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

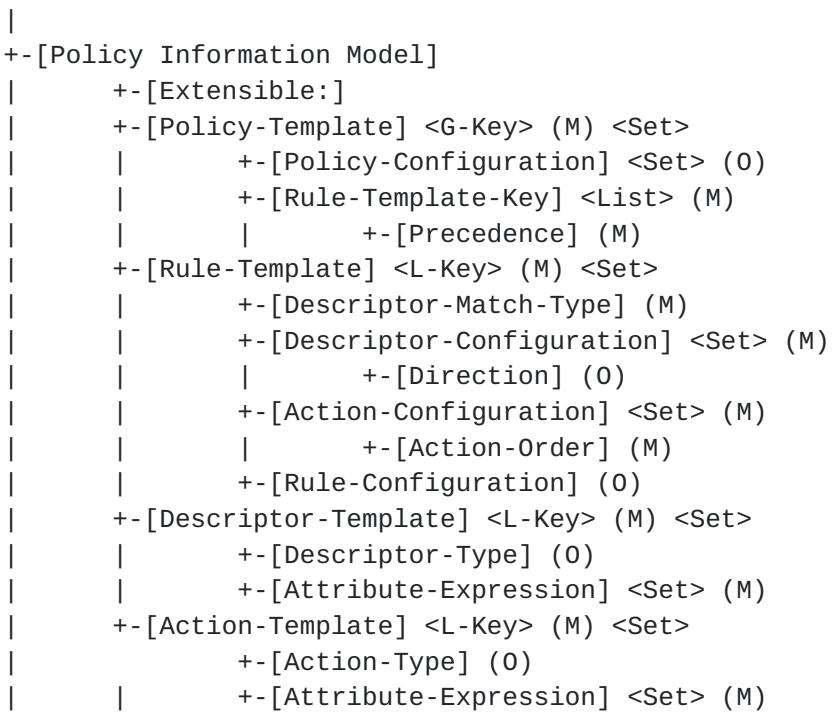

Figure 12: Policy Information Model

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

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

Policy-Key: Identifies and enables reference to this Policy definition.

Rule-Template-Key: Enables reference to a Rule template definition.

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

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

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

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

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

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

Action-Configuration: References an Action Template definition,

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

Action-Order: Defines the order in which actions are executed when the associated traffic descriptor selects the packet.

Descriptor-Template-Set: A set of traffic Descriptor Templates, each of which can be evaluated on the incoming or outgoing packet, returning a TRUE or FALSE value, defined as follows:

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

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

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

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

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

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

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

4.9.6. Mobility-Context Information Model

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

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

Figure 13: Mobility-Context Information Model

The Mobility-Context Substructure holds the following entries:

Mobility-Context-Key: Identifies a Mobility-Context

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

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

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

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

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

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

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

Domain-Key: Enables access to a Domain instance.

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

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

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

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

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

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

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

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

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

4.9.7. Monitor Information Model

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

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

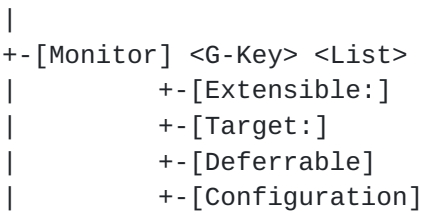

Figure 14: Monitor Substructure

Monitor-Key: Identifies the Monitor.

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

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

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

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

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

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

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

5. 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.

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.

6. IANA Considerations

TBD

7. 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.

8. References

8.1. Normative References

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

[RFC6088] Tsirtsis, G., Giarreta, G., Soliman, H., and N. Montavont, "Traffic Selectors for Flow Bindings", RFC 6088, DOI 10.17487/RFC6088, January 2011, <https://www.rfc-editor.org/info/rfc6088>.

8.2. Informative References

[I-D.bertz-dime-policygroups]
Bertz, L. and M. Bales, "Diameter Policy Groups and Sets", Work in Progress, Internet-Draft, draft-bertz-dime-policygroups-06, 18 June 2018, <http://www.ietf.org/internet-drafts/draft-bertz-dime-policygroups-06.txt>.

[I-D.ietf-dmm-deployment-models]
Gundavelli, S. and S. Jeon, "DMM Deployment Models and Architectural Considerations", Work in Progress, Internet-Draft, draft-ietf-dmm-deployment-models-04, 15 May 2018, <http://www.ietf.org/internet-drafts/draft-ietf-dmm-deployment-models-04.txt>.

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

Appendix A. 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 15: 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.

Authors' Addresses

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

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

Lyle Bertz
6220 Sprint Parkway
Overland Park KS, 66251,
United States of America

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
United States of America

Email: sgundave@cisco.com

Danny Moses

Email: danny.moses@intel.com

Charles E. Perkins
Futurewei Inc.
2330 Central Expressway
Santa Clara, CA 95050
United States of America

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