Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: May 7, 2020

E. Haleplidis

J. Hadi Salim
Mojatatu Networks

J. Chung
Viasat

November 4, 2019

ForCES-based BNG
draft-haleplidis-forces-bng-00

Abstract

This document provides an approach for the separation of the forwarding and control plane for the Broadband Network Gateway (BNG) using IETF's ForCES architecture. The document provides an initial primer on ForCES as well as an initial ForCES XML model that describes some basic functions of the BNG.

Table of Contents

1. Introduction

This document presents IETF's ForCES architecture as a basis for the control and forwarding separation for the Disaggregated Broadband Network Gateway (BNG) [TR-101].

For contextual overview, any prescribed "experience" in this document are based on deployment experience over many years at large and small deployment environments for embedded, cloud as well as data centre environments. Some of these deployments (still operational at time of writing) have been publicly hinted at in media [media1], [media2] and [media3].

2. Terminology and Conventions

2.1. Requirements Language

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

2.2. Definitions

This document reiterates the terminology defined by the ForCES architecture in various documents for the sake of clarity.

FE Model - The ForCES model used for describing resources to be managed/controlled. This includes three components; the modeling of individual Logical Functional Block (LFB model), the logical interconnection between LFBs (LFB topology), and the FE-level attributes, including LFB components, capabilities and events. The FE model provides the basis to define the information elements exchanged between CEs and FEs in the ForCES protocol [RFC5810].

LFB (Logical Functional Block) Class - A template that represents a resource that is being modeled. Most LFBs relate to packet processing in the data path; however, that is not always the case. LFB classes are the basic building blocks of the FE model.

LFB Instance - A runtime instantiation of an LFB class.

ForCES Component - A ForCES Component is a well-defined, uniquely identifiable and addressable ForCES model building block. A component has a 32-bit ID, name, type, and an optional synopsis description. These are often referred to simply as components.

ForCES Protocol - Protocol that runs in the Fp reference points in the ForCES Framework [RFC3746].

ForCES Protocol Layer (ForCES PL) - A layer in the ForCES protocol architecture that defines the ForCES protocol messages, the protocol state transfer scheme, and the ForCES protocol architecture itself as defined in the ForCES Protocol Specification [RFC5810].

ForCES Protocol Transport Mapping Layer (ForCES TML) - A layer in ForCES protocol architecture that uses the capabilities of existing transport protocols to specifically address protocol message transportation issues, such as how the protocol messages are mapped to different transport media (like TCP, IP, ATM, Ethernet, etc.), and how to achieve and implement reliability, ordering, etc. the ForCES SCTP TML [RFC5811] describes a TML that is mandated for ForCES.

Broadband Network Gateway (BNG) - is the network edge aggregation point used by subscribers as the access point through which they connect to the broadband network.

3. ForCES overview

In this section we present a quick overview of the ForCES architecture. The reader is encouraged to read the relevant documents, in particular [RFC5810], [RFC5812] and [RFC5811].

The origins of ForCES lie in the desire to separate control and datapath; where "datapath" was intended to be packet processing resources. Over time, however, due to the convenience of the ForCES architecture it has been used for controlling and managing arbitrary (other than packet processing) resources. As long as one can abstract the resources using the ForCES model, the protocol semantics allows using ForCES protocol to control and manage said resources.

In the case of the BNG, we will show later the attributes such as interfaces, user statistics and QoS parameters can all be modeled as resources.

The ForCES architecture is comprised of:

1. A data model definition [RFC5812] serving as a basis for the architecture constructs acted on by the protocol.

2. The ForCES protocol (PL) [RFC5810] which acts on the model component constructs for the purpose of control/management.

3. A transport mapping layer (TML) which takes the PL constructs and maps them to underlying convenient transport(s) and then delivers them to the target end points. Currently there is only one standardized TML based on SCTP; [RFC5811]. however more could be defined - as an example QUIC [I-D.ietf-quic-transport] appears to be a very good fit.

3.1. ForCES protocol

The ForCES protocol features can be summarized as:

1. Transport independence. The ForCES protocol is intended to run on a variety of chosen protocols.

2. Simplified ForCES layer when possible:

   * Security is left up to the transport choice keeping the ForCES layer simple.

   * Optional configurable Controller high availability. FEs(resource owners) when desired can connect to multiple controllers in both cold or hot standby mode [RFC5810], [RFC7121].

3. Degrees of reliability. Deployment experience with ForCES as depicted in the SCTP TML (RFC 5811 [RFC5811]) has shown an absolute need for a variety of shades of reliability.

4. Node overload. Deployment experience has shown the need to protect against node overload in a work-conserving mode (thus optimal resource usage).

5. Transactional capabilities in the form of 2 phase commits.

6. Wire Serialization and optimization. Encoding on the wire is binary. The data model is sufficient to describe the content on the wire.

7. Transactional scalability, low latency and high throughput.

8. Various execution modes for transactions {Execute-all-or-none, Execute-until-error, Execute-all-despite-errors}.

9. Communication methods. ForCES provides two communication methods for a controller to receive data from the device, namely request/response and publish/subscribe. ForCES allows the controller at any time to access (request) any resource, and allows for a controller to subscribe to any supported resources events.

10. Simple and powerful API. The ForCES architecture provides (very) few simple protocol verbs which act upon a multitude of nouns as represented by the ForCES model. The grammar could be described as:

    <Command> <Resource path> [Data]

    In other words, the ForCES semantic allows composing of rich services on top of the basic grammar. The expressive simplicity of the protocol is achieved due to the few verbs which act on the agreed-to modeled LFB components. The protocol is totally agnostic of the nature of the resource being controlled/managed. It is up to the modeler to describe the resource in the manner that is fitting (although frowned-upon, one could describe the resource model exactly as it is implemented and reduce the generalities and therefore translation overhead). The model is highly extensible and for this reason, the knowledge of the resource control is offloaded to the service layer and a basic infrastructure is all that is needed.

    The ForCES verbs are: {GET, SET, DELETE, REPORT and a few helpers}.

11. Traffic sensitive protocol level connectivity detection. ForCES layer heartbeats could be sent in either or both directions. Heartbeats, however are only sent when the link is idle.

12. Dynamic association of FEs to CEs. FEs register and unregister to controllers and advertise their capabilities and capacities.

3.2. ForCES Model

The ForCES Model features can be summarized as:

1. Data model modularity through LFB class definition.

2. Data model type definitions via atomic types, complex/compound types, grouping of compound types in the form of structures and indexed/keyed tables which then end up composing addressable components within an LFB class.

3. Hierarchical/tree definition of control/config/state components which is acted on by a controller via the ForCES protocol.

4. Information-modeled metadata and expectations.

5. Built in LFB class resource capability definition/advertisement.

6. Publish/subscribe LFB event model with expressive trigger and report definitions. Filters include:

   * Hysteresis - used to suppress generation of notifications for oscillations around a condition value. Example, "generate an event if the link laser goes below 10 or goes above 15".

   * Count - used to suppress event generation around occurrence count. Example, "report the event to the client only after 5 consecutive occurrences".

   * Time interval - used to suppress event generation around a time limit. Example, "Generate an event if table foo row hasn't been used in the last 10 minutes".

7. Data model flexibility/extensibility through augmentations, and inheritance.

8. Backward and forward compatibility via LFB design and versioning rules.

9. Formal constraints for validation of defined components.

An LFB class can be seen in Figure 1. An LFB class has configurable components, read-only capabilities and subscribable events. Also an LFB class has one or more input ports where packets and/or metadata enter the LFB, and one or more output ports where packets and/or metadata exit the LFB. LFBs can be instantiated in the datapath.

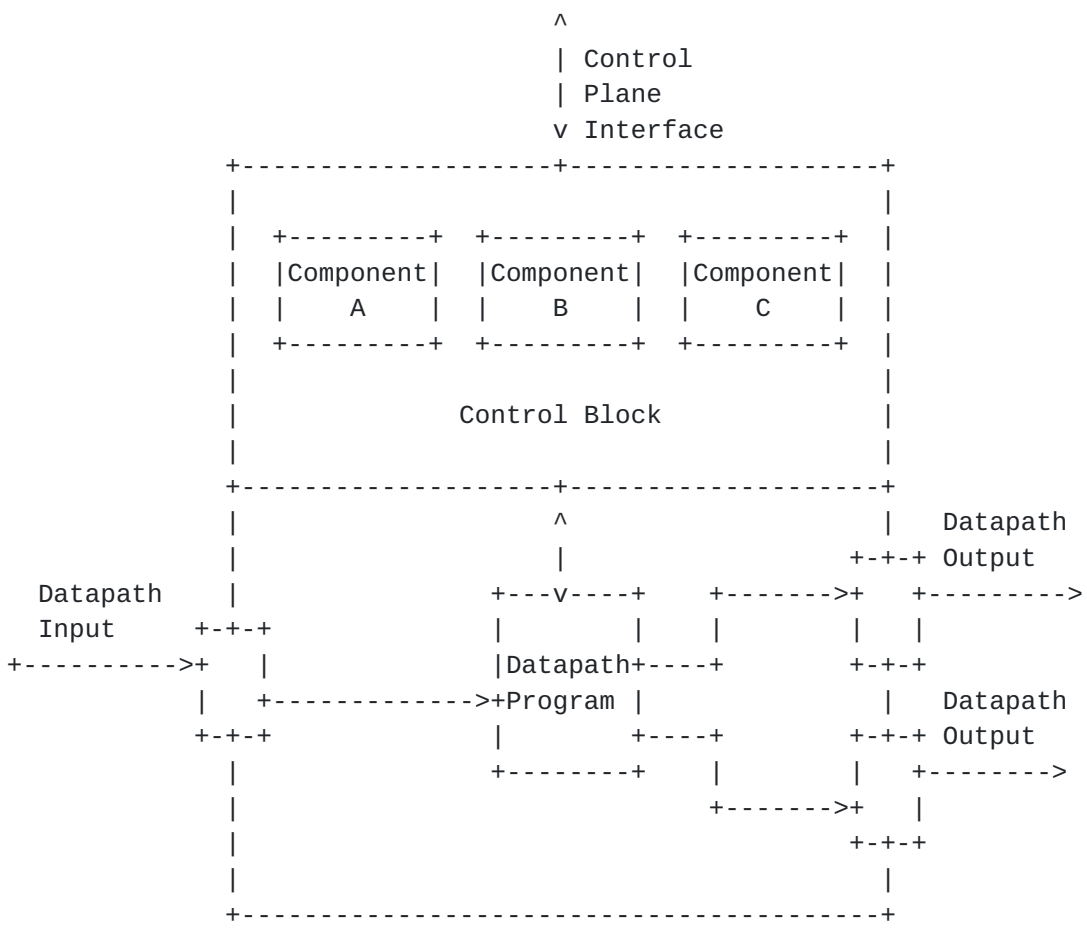

Figure 1: LFB Model

Most LFBs are related to packet processing. However there are cases such as discussed in [RFC5810] and [RFC5812] where two LFBs are defined, the FE Protocol LFB and the FE Object LFB respectively. The FE Protocol LFBs allows the control plane to configure, among others, the ForCES protocol mechanics, such as High availability and heartbeat mechanism. The FE Object LFB allows the control plane to configure, among others, the FE as whole, instantiate LFBs and manipulate the LFB service graph respectively.

3.3. ForCES & BNG

The ForCES architecture accrues several benefits. ForCES has the ability to add new packet services, described as LFB graphs, in an existing infrastructure. Secondly it can natively support any type of access, fixed, mobile, simply by modeling the necessary LFBs. One of the major advantages is that these abilities come with no change to the protocol; so long as the proper models (LFBs) are introduced, the ForCES protocol inherently supports them. To illustrate these claims, this document will start first by describing a basic

connectivity service and then augment that with a bandwidth management service.

For more details on how ForCES can support the separation of the forwarding from the control plane in regards to the BNG, the reader is encouraged to read the Control-Plane and User-Plane (CUSP) ForCES gap analysis [I-D.haleplidis-bcause-forces-gap-analysis] which elaborates on how ForCES meets the CUSP requirements as well as provide a ForCES XML model that detailed a CUSP information model. We hope to convince the reader that there already exists a robust IETF architecture which has a large deployment experience that meets all the CUSP requirements, while being more extensible and flexible with new requirements without any changes to the protocol.

4. Basic BNG ForCES model

The BNG is the network edge aggregation point used by subscribers as the access point through which they connect to the broadband network. A subscriber could get access to the network using one or more different access types through the BNG. In this draft we focus on IPv4 PPPoE access.

Supporting different access types is easily done by adding appropriate LFBs that handle encapsulating and decapsulating the relevant protocol headers. As discussed in the previous section, adding new LFB models has no impact on the protocol itself. Future versions of this document will include additional access types.

A BNG is comprised of many different components. Figure 2 depicts components, modeled through ForCES, required to provide subscriber access using PPPoE. It is important to note that Figure 2 does not provide implementation details, but rather is a modelling of the underlying resources.

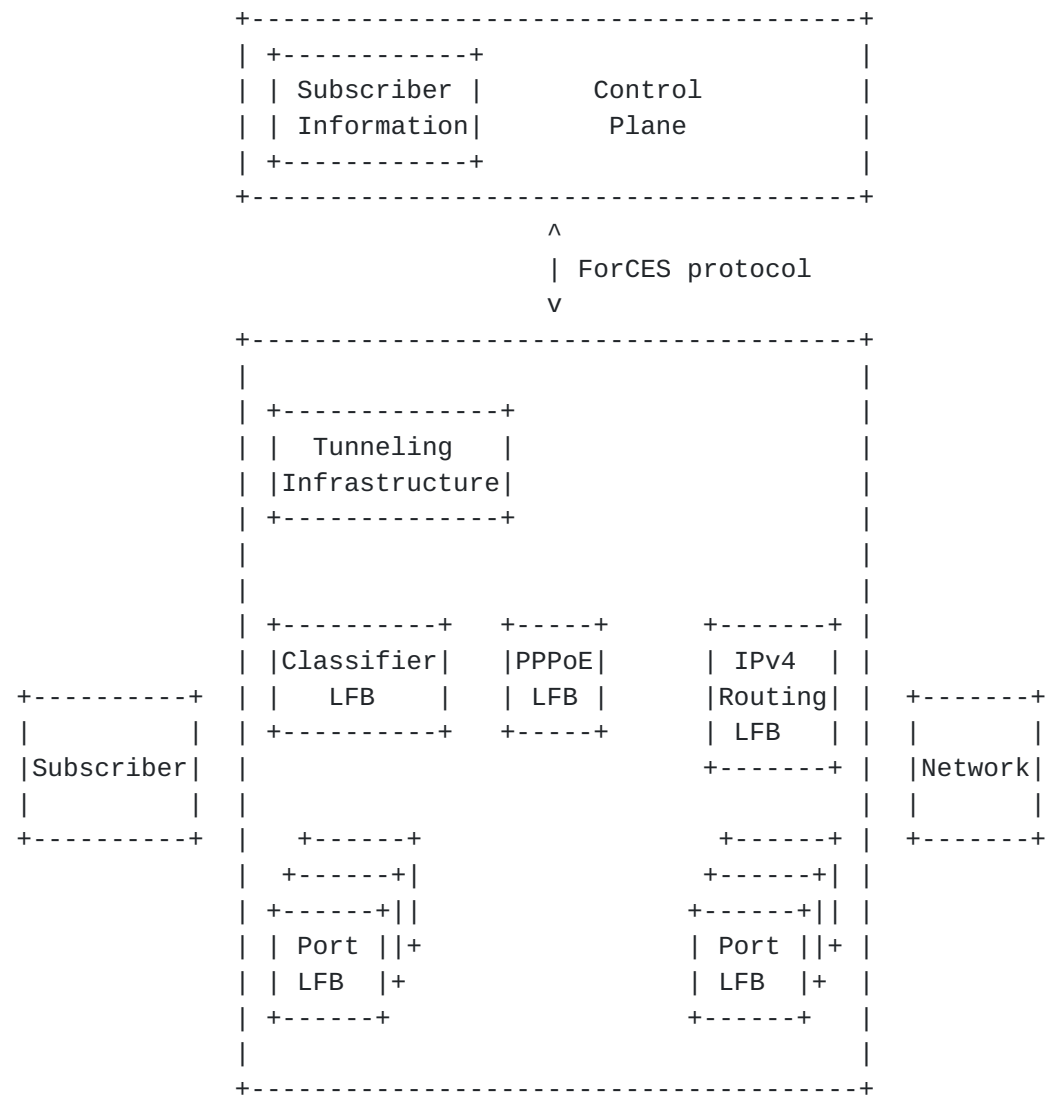

Figure 2: BNG LFB Components

There are a number of steps required for a subscriber to get access to the network. These include:

1. First the subscriber has to be authenticated. This will require a number of control packets to be exchanged between the subscriber and the control plane, redirected from the forwarding plane. For PPPoE the credentials are handed off to a RADIUS server; The communication between the control plane and the RADIUS server is out of scope of this document.

2. Once the subscriber is authenticated, the control plane has to configure the forwarding plane with the subscriber information, including resource allocation, authorizing the subscriber by providing access control and subscribed services.

3. Finally, as subscriber traffic is allowed through the BNG, the appropriate LFB instances are monitored by the control plane for accounting purposes either by polling or subscription to events. This would require either the control plane to poll the BNG or subscribe to accounting events from appropriate LFB instances.

In the first version of the draft, we focus on PPPoE as the access control mechanism. This first stage is the discovery and authentication phase via PPPoE and PPP.

4.1. Authentication

There are two stages in PPPoE [RFC2516]. The Discovery stage and the PPP Session stage that will perform the discovery and authentication respectively.

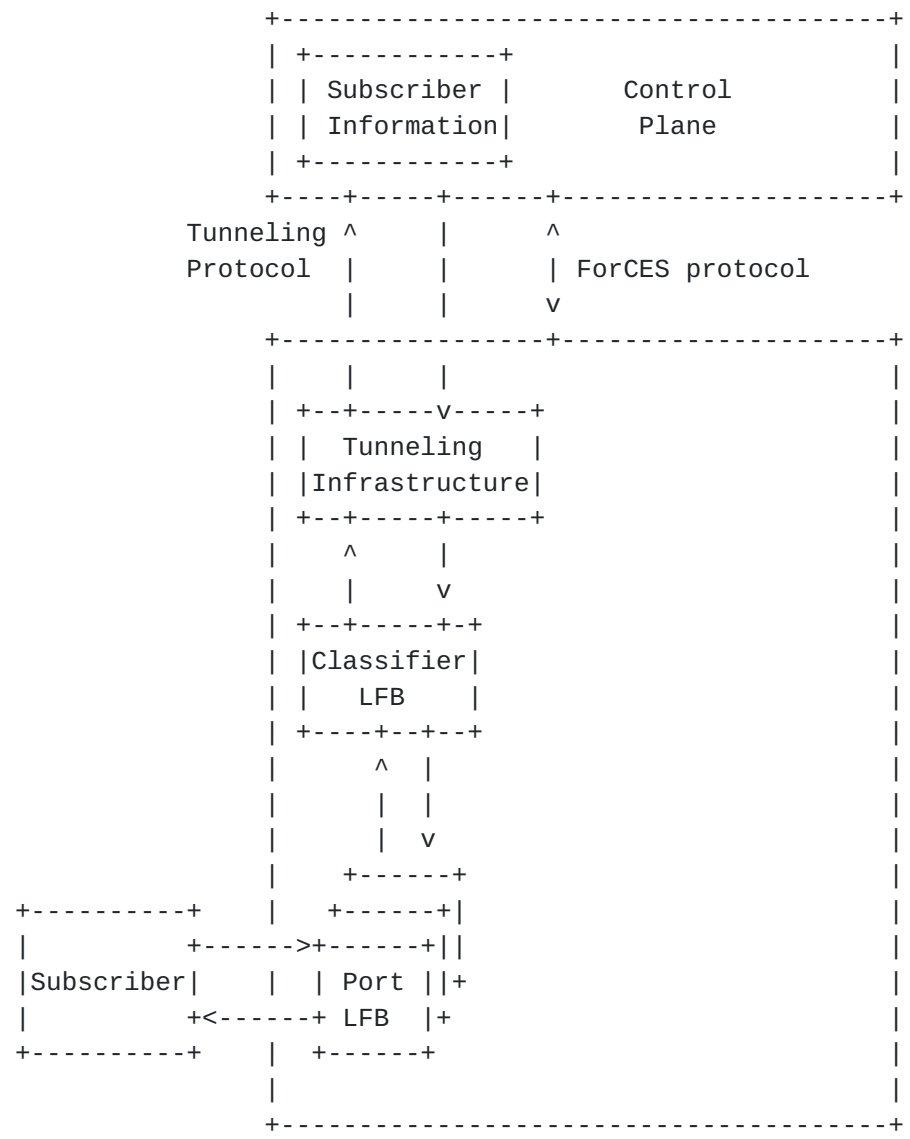

Figure 3: BNG Control Traffic handling

4.1.1. PPPoE Discovery stage

During the Discovery stage, frames arrive at an Ethernet Port (Port LFB). The frames are sent to the Classifier LFB as seen in Figure 3. The Classifier LFB determines whether these frames are control packets, distinguished by ethertype 0x8863. The control frames will be redirected via a tunneling infrastructure towards the control plane.

The tunneling infrastructure could be implemented by VxLAN, GRE, etc. ForCES could also be used in the case to redirect packets from the forwarding plane to the control plane. This is currently out of scope of this document.

The control plane handles the control messages and sends back the responses via the tunneling infrastructure to be redirected in the data path to be sent back to the subscriber.

Several control messages are exchanged in this stage between the subscriber and the controller - all recognized by the Classifier LFB, based on ethertype value 0x8863, and redirected to the control plane.

At the end of the PPPoE discovery stage both peers know the Session ID and the peer's MAC address which both identify uniquely the session.

At this stage the control plane knows the subscriber's Session ID and MAC address and programs the classifier.

4.1.2. PPP Session stage

After the PPPoE Discovery stage has ended PPP traffic starts tp flow. Frames arrive at the ethernet Port (LFB). The frames are sent to the Classifier LFB. PPPoE data packets are now distinguished by ethertype 0x8864 but there are still control packets passing prior to the subscriber being authenticated.

The Classifier LFB will distinguish control vs data packets, based on the protocol field of the encapsulated PPP header. Control packets such as LCP (0xC021), PAP (0xC023), CHAP (0xC223) and IPCP (0x8021), will be sent to the Control Plane to authenticate and authorize the Subscriber.

After the subscriber has been authenticated the authentication stage is finished, an IP address will be issued and the Subscriber will be considered authenticated and authorized.

4.2. Subscriber Information Configuration

At this stage the controller is ready to configure the forwarding plane with the authorized subscriber information.

The control plane will create the basic connectivity service by configuring the PPP and the IPv4 Routing LFB to operate in forwarding mode in both the subscriber side direction as well as the internet side directions. The control plane will send a number of ForCES messages via the ForCES protocol to these LFBs with the parameters specific for the subscriber.

The Subscriber provisioned information, such as assigned IP, Session ID, service, MAC address, can reside either in the CE or in the FE as an individual LFB. Both options are valid in ForCES case. The

subscriber information, once set, will trigger a set of control commands to a number of LFBs; having it in the Control Plane will require more commands on the wire (ForCES messages) to be sent to the different LFBs. On the other hand, if the Subscriber Information rests in the data plane then less messaging is needed between control and data plane.

Once the subscriber information has been configured correctly, the subscriber's traffic starts passing through as illustrated in Figure 4. The Classifier LFB will distinguish PPPoE data packets by ethertype 0x8864, session ID, and IPv4 traffic by PPP protocol header 0x0021 and send them to the PPPoE LFB (with session ID as metadata) on the DecapIn input port of the PPPoE LFB.

The PPPoE LFB, once it validates the packet has arrived from an authorized subscriber based on the session ID and the MAC address, will decapsulate the IPv4 packet and send it to the IPv4 routing LFB, via the DecapOut output port.

The IPv4 routing LFB will route the packets and send it out the correct port LFB on the other side to be sent into the network.

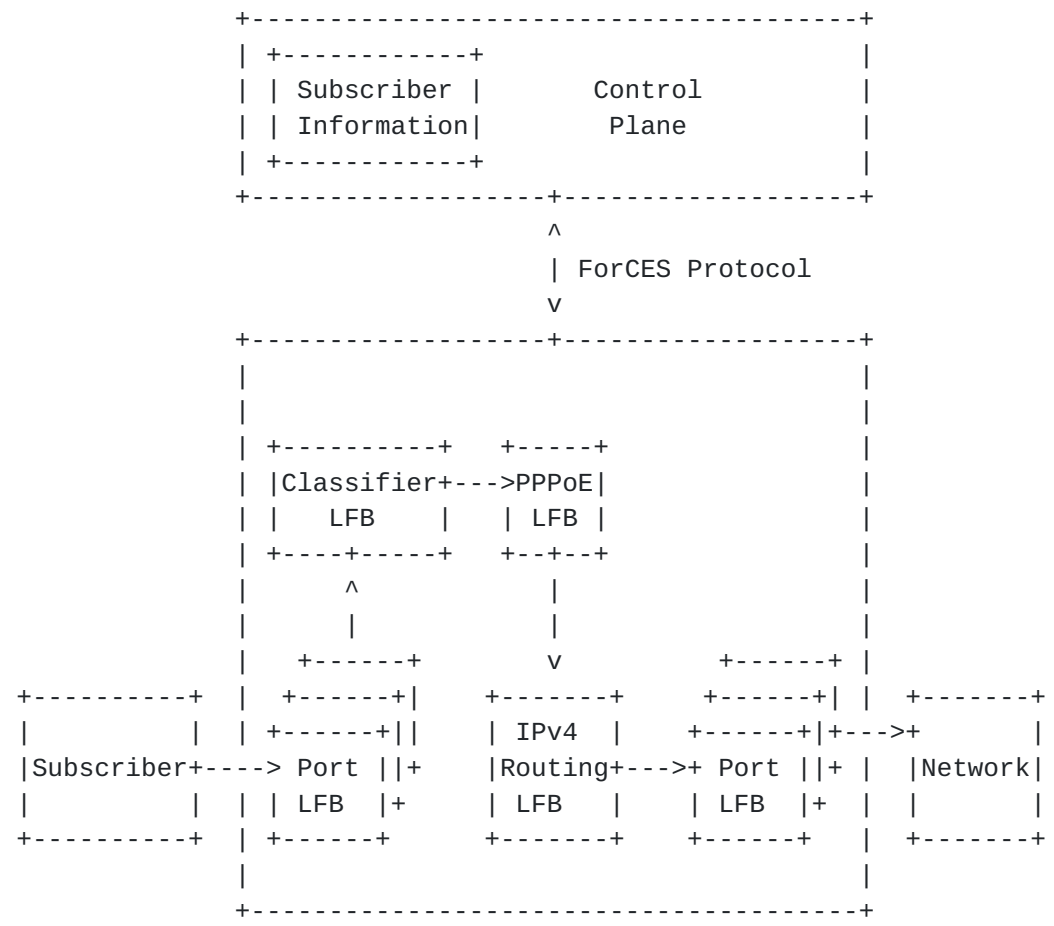

Figure 4: BNG Basic Connectivity Service (Upstream)

Figure 5 shows the downstream processing. When a packet is received from the network the port LFB will send the packet to the classifier LFB.

The classifier LFB will select the IPv4 traffic and generate a metadata (Subscriber ID) based on the destination IP address, to be propagated to downstream LFBs, and send the packets to the IPv4 Routing LFB. The IPv4 Routing LFB perform normal routing functions, such as selecting the correct output port and decreasing TTL. and pass on the packet to the PPPoE LFB through the EncapIn port.

The PPPoE LFB will encapsulate the packet with the correct PPPoE Session ID and destination MAC address and send the traffic back to the subscriber via the EncapOut port to the Port LFB with the interface index provided by the IPv4 routing LFB.

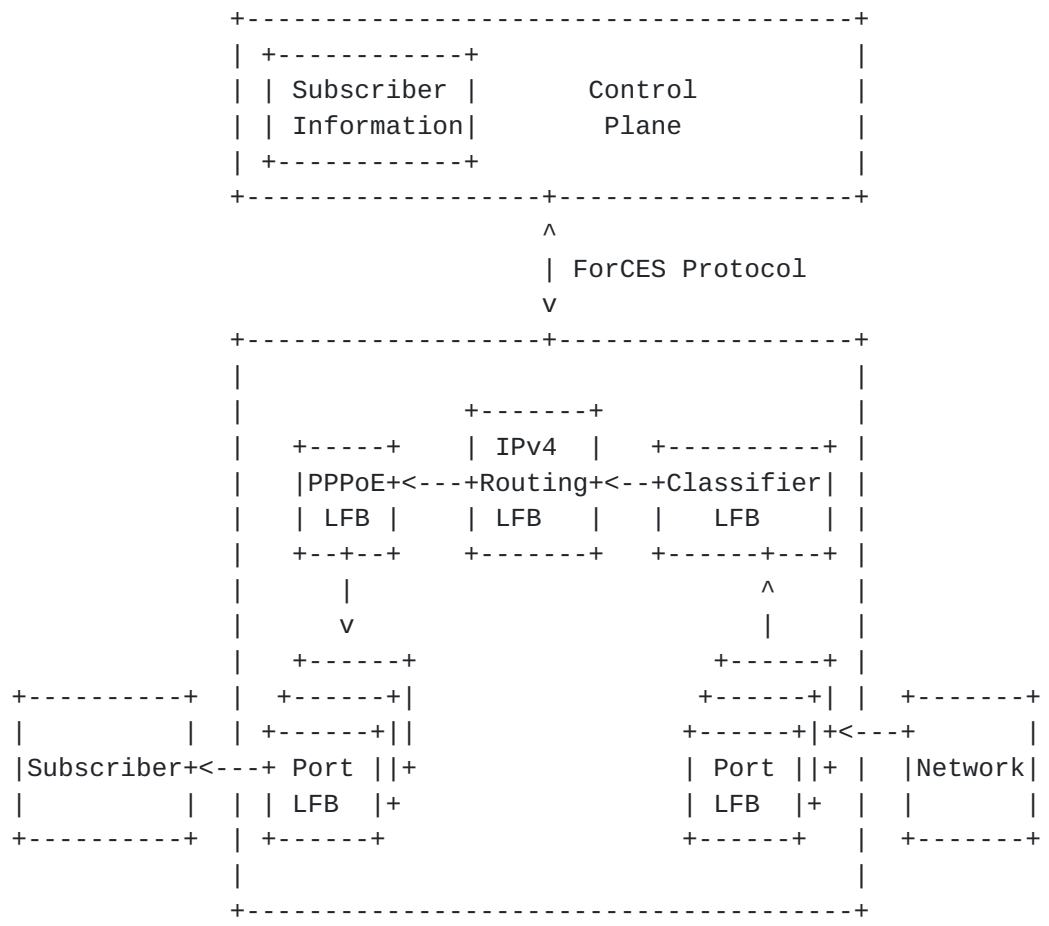

Figure 5: BNG Basic Connectivity Service (Downstream)

4.2.1. Supporting multiple access types

As has been discussed earlier in the document, any kind of access mechanism can be supported by the ForCES model, whether that is fixed access, or mobile or wireless, or optical.

A relevant port LFB instantiation will be needed to handle traffic entering the BNG

The classifier can be augmented to distinguish the packets based on their access mechanism, e.g. ethertypes for ethernet based access types and will send the packets to the relevant LFBs to be handled.

This has no effect on the ForCES protocol. So long as everything has been modeled with the ForCES model, the protocol will able to control and manage the LFBs with no change.

4.3. Traffic monitoring

As soon as Subscriber traffic starts flowing through the BNG, it is imperative to monitor for accounting purposes.

Subscriber usage monitoring can be achieved in a number of ways, with the easiest being monitoring incoming and outgoing statistics at the PPPoE LFB. The controller can either poll the LFB for said statistics, or it can subscribe to events to the PPPoE LFB and receive notifications for statistics.

5. Advanced BNG Services

The BNG is composed of many more functions and components. ForCES provides the ability to provision new services.

A service, in ForCES terms, is a graph of LFBs. Figure 4 and Figure 5 showcase a simple connectivity service. The operator via the control plane can provision new services by creating new graphs with existing LFBs or introducing new LFBs and composing new LFB graphs. One important detail in ForCES, as has been discussed before, is that defining or adding new LFBs has absolutely no impact on the protocol itself.

5.1. Bandwidth Management Service

A service provider can introduce a bandwidth management service with no extension to the protocol. Adding rate limiting both in the upstream to the internet and downstream from the internet is simply done by adding a new LFB called policer as can be seen in Figure 6 and Figure 7. A service provider could add an instance for each subscriber to give them different bandwidth rates based on SLAs. Example 1 mbps up/3 mbps down vs 3 mbps up and 8mbps down.

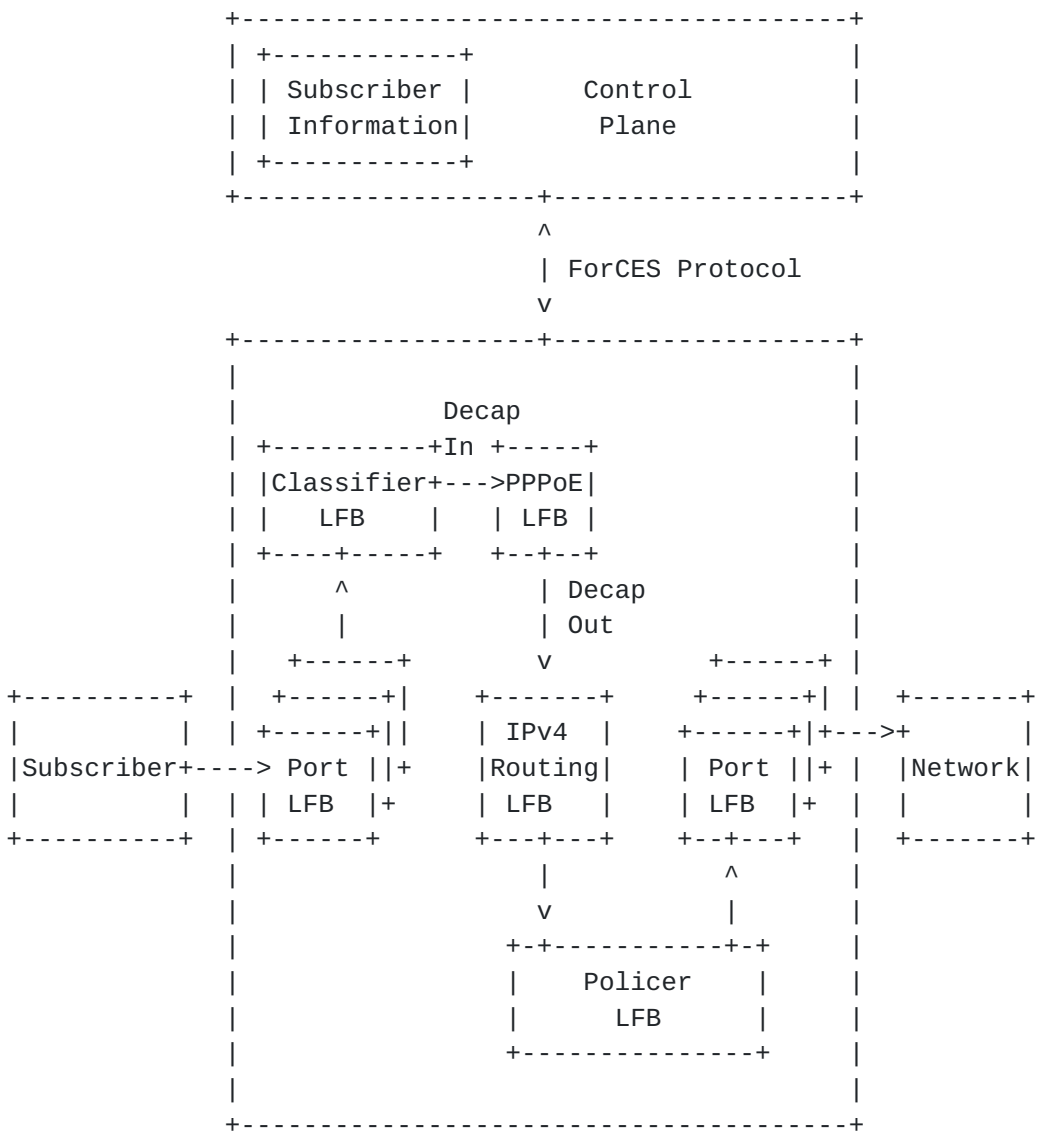

Figure 6: BNG Bandwidth management service (Upstream)

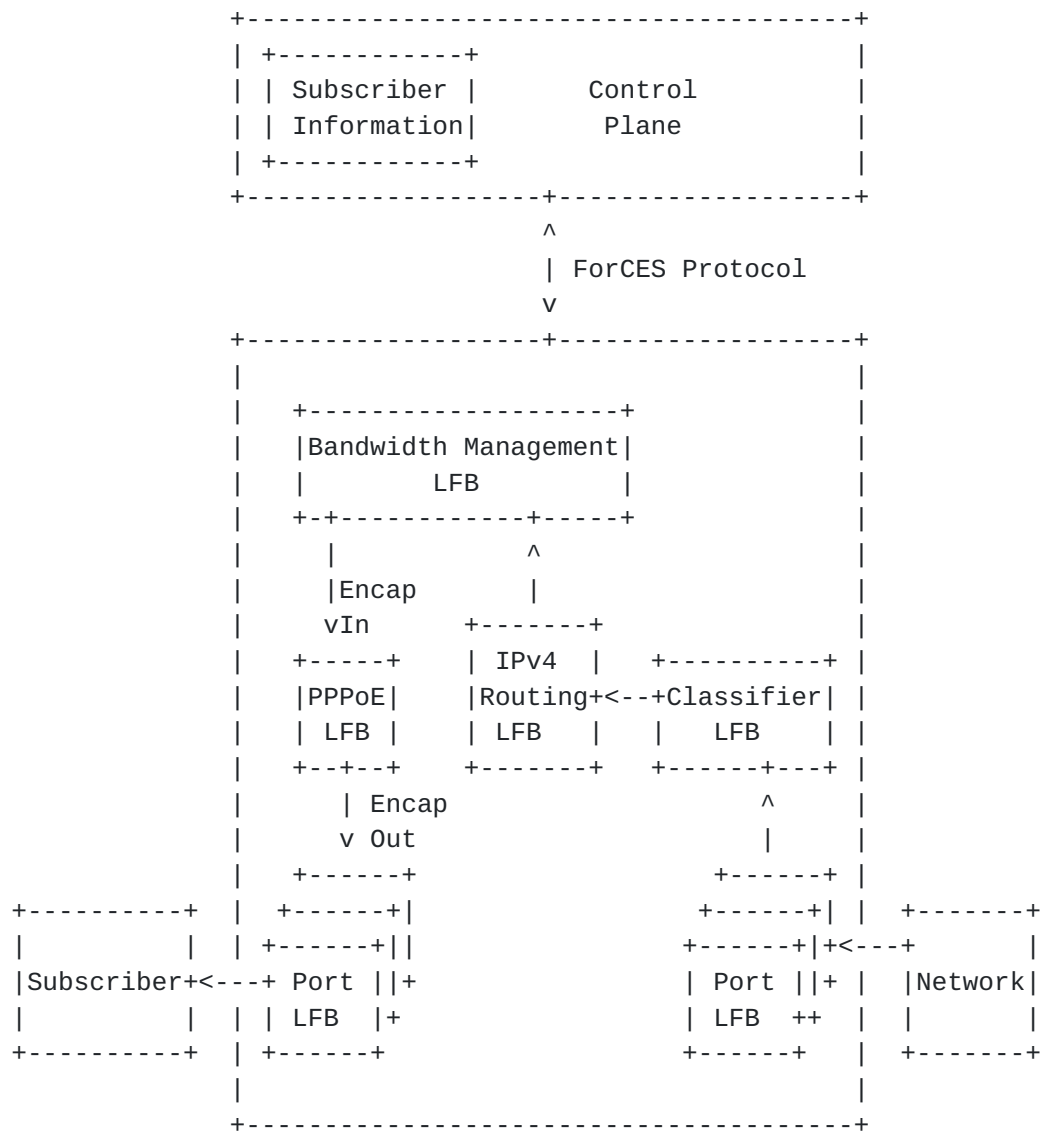

Figure 7: BNG Bandwidth management service (Downstream)

5.2. Stateless access control service

In case an operator requires services like a firewall, an Access Control List (ACL) LFB can be instantiated and inserted into the graph to drop or allow packets.

5.3. Quota Enforcement service

In case an operator require to perform quota enforcement, a Quota Enforcement (QE) LFB can be instantiated and inserted into the graph to drop or allow packets depending on the Subscriber SLA.

5.4. Lawful Intercept service

In the case of creating a Lawful Intercept, an operator simply by instantiating a mirror LFB into the datapath to create copies of the packets to be sent to a specific destination

6. LFB Class Descriptions

As has been discussed before, an LFB is a well defined logical functional block residing in the FE and is an accurate abstraction of the FE's processing capabilities.

This document provides a sample LFB class library pertaining to the separation of the forwarding and the control plane for the BNG. [RFC6956] provides an additional LFB class library for a typical router.

6.1. Port LFB

The Port LFB abstracts all the interfaces, physical and virtual, in the device. The LFB handles frames coming in from or out of the FE. The current port class LFB is defined to receive ethernet frames.

6.1.1. Data Handling

When frames arrive from the Subscriber or the Network side, these frames are received by the Port LFB via IngressInPort, the physical port of the BNG, to be passed downstream to the Classifier LFB via the IngressOutPort.

When frames are meant to be sent to the Subscriber or the Network side, these frames arrive to the Port LFB via the EgressInPort alongside an ifIndex metadata to notify which interface should be used, and sent out via the EgressOutPort.

Figure 8 illustrates the Port LFB

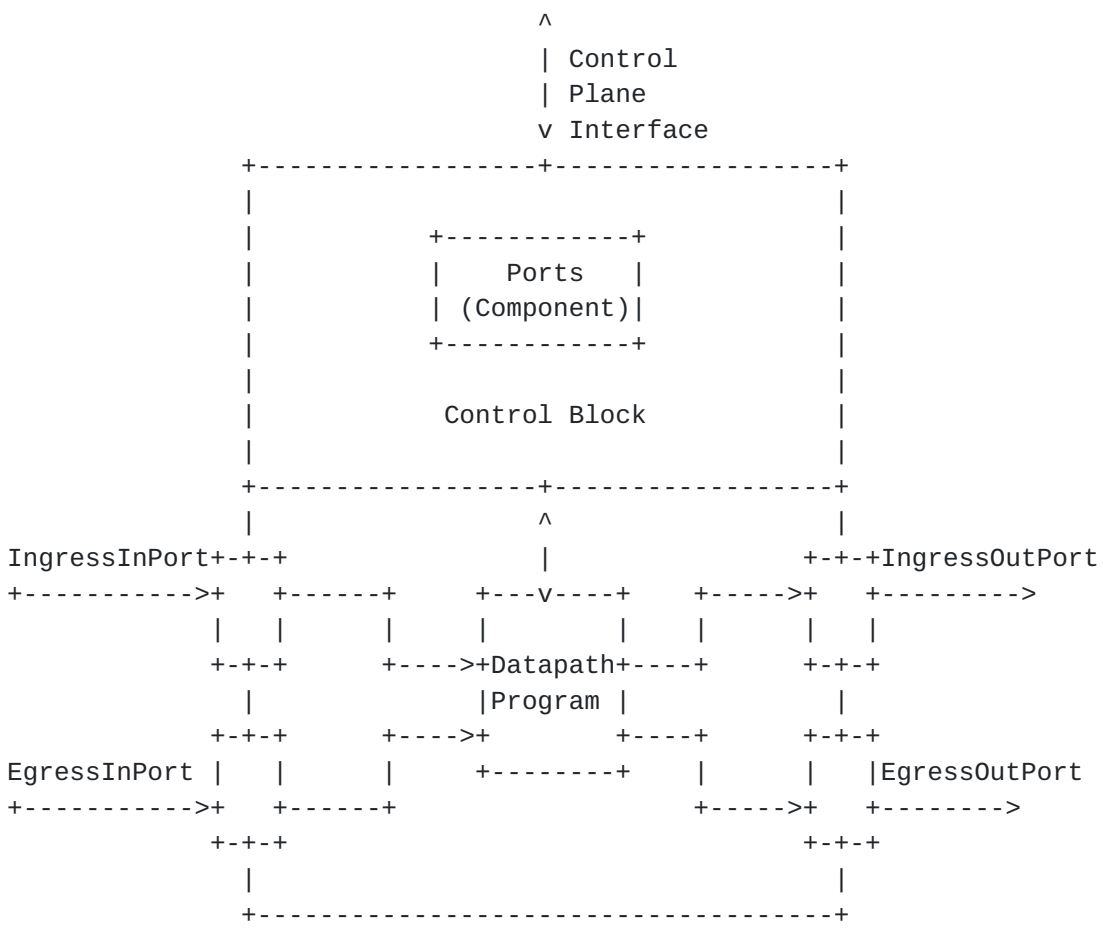

Figure 8: Port LFB

6.1.2. Components

This LFB has defined only one component, named ports. The ports component is table. Each entry in the table contains the parameters for a port within the BNG. All the entries consist of all the ports of the BNG. Each table entry is of type PortInfo.

The PortInfo type contains the following configurable parameters:

Name - A string representation for the name of the port

ifindex - the interface index of the port

L2Address - The MAC address of the port

MTU - The maximum transmit unit for this port

Stats - 64-bit counters for statistics for the port. These include Tx,Rx bytes and packets, errors and dropped packets.

Operstate - The Operational state of the port, such as down, up

Adminstate - The Administrative state of the port, such as down, up

Promiscuity - The port promiscuity

6.1.3. Capabilities

This LFB does not have a list of capabilities.

6.1.4. Events

This LFB has four events specified:

Port changed - This event will trigger when an existing port has been updated.

Port deleted - This event will trigger when an existing port has been deleted such as deleting a virtual port or removing a physical interface.

Port created - This event will trigger when an new port has been created such as creating a virtual port or inserting a physical interface.

Port stats changed - This event will trigger periodically when stats change. It is important to properly configure the event parameters (hysteresis, interval) to avoid generation of multiple event notifications

6.2. Classifier LFB

The classifier LFB abstracts the classification of packets into different categories. From the subscriber side to the network, it is required for the classifier LFB to detect Subscriber control traffic to be redirected to the control plane, as well as detect Subscriber data traffic to be sent downstream with the subscriber ID as metadata. From the network side to the subscriber, it is required for the classifier LFB to detect the Subscriber ID from the destination IP address and send the packet with the subscriber ID as metadata to the next LFB. In addition the classifier LFB has to discriminate control packets arriving from the control plane via the tunneling infrastructure which have to be sent back to the subscriber.

6.2.1. Data Handling

Packets enter the classifier LFB via the InPort. Ethernet packets arriving from the subscriber side with Ethertype 0x8863 are considered PPPoE control messages and are sent to the control plane via the RedirectOut output port. Similarly, ethernet frames with EtherType 0x8864 but the PPP protocol field belonging to a control protocol, will be sent to the control plane via the ControlOut output port

Packets returning via the tunneling infrastructure enter the classifier LFB via the InPort, are matched based on destination MAC address, Ethertype and optionally PPP protocol field and are sent to the Port LFB via the EthernetOut port to be sent to the subscriber.

Subscriber data traffic coming from the subscriber's side are matched based on EtherType 0x8864 and PPP protocol 0x0021 (IPv4) and also based on the PPPoE session ID and Subscriber MAC address. Once matched, the classifier will issue the Subscriber ID (configured by the control plane) and pass it along the packet as metadata to the next LFB (PPPoE LFB) via the EthernetOut port.

Subscriber data traffic coming from the network's side are matched based on IPv4 destination address. Once matched, they will be issued the Subscriber ID (configured by the control plane) and pass it along to the next LFB (IPv4 Routing) via the IPv4Out port.

6.2.2. Components

This LFB has one component, the filters which is the table that contains all the classification filters.

Each entry in the table contains three fields:

- Filter Name - The name of the filter
- Actions - A table of actions to be performed on the packet
- Filter keys - A table of key values to be matched on the packets

In regards to the Actions, each table entry of the actions contains two fields:

- Action Type - The name of the action
- Actions - The action itself. Currently only three actions have been defined, drop forward and redirect to the controller

In regards to the Filter keys, each table entry of the filter keys contains five optional fields:

IP protocol - Detects a specific transport protocol

Ethernet - Matches on specific ethernet fields

IP - Matches on specific IPv4 header fiels

PPP Control - To detect and match specific PPP control messages

PPP Subscriber traffic - To detect PPP subscriber data traffic

6.2.3. Capabilities

This LFB does not have a list of capabilities.

6.2.4. Events

This LFB has three events specified:

Filter Changed - An Existing filter has been changed.

Filter deleted - This event will trigger when an existing filter has been deleted.

Filter created - This event will trigger when a new filter has been created.

6.3. PPPoE LFB

The PPPoE LFB is responsible for Encapsulating IPv4 Packets into PPPoE Ethernet frames coming from the network side towards the subscriber, and decapsulating IPv4 Packets from PPPoE frames arriving from the subscriber to be sent to the network.

6.3.1. Data Handling

This LFB handles traffic different depending on whether it needs to be encapsulated into an PPPoE frame or decapsulate a PPPoE frame and the IPv4 packet from within.

Regarding encapsulation, IPv4 packets arrive from the network side via the EncapIn port alongside the subscriber ID generated by the classifier LFB and the OutIfIndex metadata generated by the IPv4 routing LFB. The LFB looks up the subscriber information and creates the PPPoE frame based on the information located in the PPPoE Subscriber Information table entry. The generated frame is sent to

the Port LFB via the EncapOut port to the port specified by the OutIfIndex.

Regarding decapsulation, PPPoE frame arrive from the subscriber side via the DecapIn port alongside the subscriber ID generated by the classifier LFB. The LFB looks up the subscriber information and decapsulates the IPv4 packet within. The resulting IPv4 packet is sent to the IPv4 Routing LFB via the DecapOut port.

6.3.2. Components

This LFB has two components, the first is the PPPoE Subscriber Information and the second are the Encap and Decap Statistics for encapsulating and decapsulating packets. Both components are tables.

Each entry in the PPPoE Subscriber Information table contains the following:

Subscriber ID - The subscriber identifier. This value is also used to look up table entries.

PPPoE Session ID - The Session ID of the Subscriber

Subscriber MAC Address - The Subscriber's MAC Address

Local MAC Address - The MAC Address of the local interface connected to the Subscriber.

MSS - The Maximum Segment Size

MRU - The Maximum Receive Unit

Magic Number - The magic number

Peer Magic Number - The subscriber's magic number

Each entry in the Statistics table contains the following:

Subscriber ID - The subscriber identifier. This value is also used to look up table entries.

Rx Encap Packets - The number of packets received at the encap ingress side

Tx Encap Packets - The number of packets transmitted at the encap egress side

Rx Encap Bytes - The number of bytes received at the encap ingress side

Tx Encap Bytes - The number of bytes transmitted at the encap egress side

Rx Decap Packets - The number of packets received at the decap ingress side

Tx Decap Packets - The number of packets transmitted at the decap egress side

Rx Decap Bytes - The number of bytes received at the decap ingress side

Tx Decap Bytes - The number of bytes transmitted at the decap egress side

6.3.3. Capabilities

This LFB does not have a list of capabilities.

6.3.4. Events

This LFB has four events specified:

Subscriber Changed - Existing PPPoE subscriber information has been changed.

Subscriber deleted - This event will trigger when an existing PPPoE subscriber information has been deleted.

Subscriber created - This event will trigger when a new PPPoE subscriber information has been created.

Stats changed - This event will trigger periodically when stats change. It is important to properly configure the event parameters (hysteresis, interval) to avoid generation of multiple event notifications

6.4. IPv4 Routing LFB

The model of IPv4 Routing LFB is responsible for selecting the next hop for upstream and downstream traffic and then to generate the correct MAC address to be inserted into the destination MAC address and the ifindex of the output port so that the packet can be send out of the BNG.

6.4.1. Data Handling

This LFB is used for packets sent from the Subscriber to the Network and back to locate the next hop and select the output port. IPv4 packets enter this LFB via the InPort, alongside the Subscriber Index as metadata. The IPv4 Routing LFB performs a lookup using the destination IP address on the IPv4Routing table. Once an entry has been found, the packet is checked for MTU and the next hop index is found. The next hop index is used to locate the correct enty in the IPv4NextHopTable. Once the entry has been found, the IPv4 Routing LFB decrements the TTL and populates the destination MAC address and sends the packet to the Port LFB with the ifindex metadata specifying which output port is going to be used via the NormalOut port. If an error occurs in the checksum, or TTL, packets are dropped and exit the LFB via the ExceptionOut port.

6.4.2. Components

This LFB has two components, the first is the IPv4Routing Table which is the table for the IPv4 Longest Prefix Match and the second is the IPv4NextHopTable which contains information required for the next hop. Both components are tables.

Each entry in the IPv4Routing table contains the following:

Subscriber ID - The subscriber identifier.

IPv4 Address - The destination IPv4 address which is used to look up table entries.

Prefix Length - The prefix length

Hop Selector - Index for the Next Hop Table to lookup the Next hop information.

Each entry in the IPv4NextHopTable table contains the following:

OutIfIndex - This is the interface index out of which this packet should be sent.

MTU - The MTU of the outgoing port.

Next Hop IP Address - The next hop IPv4 Address

Next Hop MAC Address - The next hop MAC Address.

6.4.3. Capabilities

This LFB does not have a list of capabilities.

6.4.4. Events

This LFB does not have a list of events.

6.5. Policer LFB

The Policer LFB is responsible for maintaining the QoS requirements for a specific user based on his SLA either for upstream or for downstream traffic. Currently only four characteristics have been defined. The four characteristics are the Committed Information Rate, the Peak Information Rate, the Committed Burst Size and the Peak Burst Size. This LFB can be further augmented with more QoS parameters.

6.5.1. Data Handling

For upstream this LFB receives IPv4 packets and Subscriber ID as metadata at the InUpstreamPort and based on the QoS parameters for that specific Subscriber, manipulates the packets to maintain the agreed upon SLA. Output packets are sent out via the OutUpstreamPort alongside the metadata, Subscriber ID.

Similarly for downstream traffic, this LFB receives IPv4 packets and Subscriber ID as metadata at the InDownstreamPort and based on the QoS parameters for that specific Subscriber, manipulates the packets to maintain the agreed upon SLA. Output packets are sent out via the OutDownstreamPort alongside the metadata, Subscriber ID.

6.5.2. Components

This LFB has been defined with two components, one for upstream and one for downstream traffic named PoliceUpstream and PoliceDownstream respectively. Both components are tables. Each entry in each table contains the four QoS parameters and the Subscriber ID. Each table entry is of type PolicerTableEntry.

The PolicerTableEntry type contains the following configurable parameters:

SubscriberID - A uint32 value to identify the Subscriber

CIR - a uint32 value to represent the Committed Information Rate in kilobits per second.

PIR - a uint32 value to represent the Peak Information Rate in kilobits per second.

CBS - a uint32 value to represent the Committed Burst Size in bytes.

PBS - a uint32 value to represent the Peak Burst Size in bytes.

6.5.3. Capabilities

This LFB does not have a list of capabilities.

6.5.4. Events

This LFB does not have a list of events.

7. BNG ForCES XML model

In this section we provide ForCES based XML models for the LFBs in the BNG ForCES model

```
<?xml version="1.0" encoding="UTF-8"?>
<LFBLibrary xmlns="urn:ietf:params:xml:ns:forces:lfbmodel:1.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
provides="vbng">
  <frameDefs>
    <frameDef>
      <name>Arbitrary</name>
      <synopsis>Any kind of packet</synopsis>
    </frameDef>
    <frameDef>
      <name>EthernetFrame</name>
      <synopsis>An ethernet frame</synopsis>
    </frameDef>
    <frameDef>
      <name>IPv4Packet</name>
      <synopsis>An IPv4 packet</synopsis>
    </frameDef>
  </frameDefs>
  <dataTypeDefs>
    <!-- For Port LFB -->
    <dataTypeDef>
      <name>operstates</name>
      <synopsis>
        The possible operational states of a port link (RFC 2863)
      </synopsis>
      <atomic>
        <baseType>uchar</baseType>
```

```
        <specialValues>
          <specialValue value="0">
            <name>OS_UNKNOWN</name>
            <synopsis>Unknown value</synopsis>
          </specialValue>
          <specialValue value="1">
            <name>OS_NOTPRESENT</name>
            <synopsis>The link is not present</synopsis>
          </specialValue>
          <specialValue value="2">
            <name>OS_DOWN</name>
            <synopsis>The link is operationally down</synopsis>
          </specialValue>
          <specialValue value="3">
            <name>OS_LOWERLAYERDOWN</name>
            <synopsis>The link of the lower port is down</synopsis>
          </specialValue>
          <specialValue value="4">
            <name>OS_TESTING</name>
            <synopsis>The Link is undergoing some testing</synopsis>
          </specialValue>
          <specialValue value="5">
            <name>OS_DORMANT</name>
            <synopsis>Link is in the dormant state</synopsis>
          </specialValue>
          <specialValue value="6">
            <name>OS_UP</name>
            <synopsis>The Link is operationally up</synopsis>
          </specialValue>
        </specialValues>
      </atomic>
    </dataTypeDef>
    <dataTypeDef>
      <name>adminstates</name>
      <synopsis>
        The possible administrative states of a port link (RFC 2863)
      </synopsis>
      <atomic>
        <baseType>uchar</baseType>
        <specialValues>
          <specialValue value="1">
            <name>AS_DOWN</name>
            <synopsis>The link is operationally down</synopsis>
          </specialValue>
          <specialValue value="2">
            <name>AS_LOWERLAYERDOWN</name>
            <synopsis>The link of the lower port is down</synopsis>
          </specialValue>
```

```
          <specialValue value="3">
            <name>AS_DORMANT</name>
            <synopsis>Link is in the dormant state</synopsis>
          </specialValue>
          <specialValue value="4">
            <name>AS_UP</name>
            <synopsis>The Link is operationally up</synopsis>
          </specialValue>
        </specialValues>
      </atomic>
    </dataTypeDef>
    <dataTypeDef>
      <name>devstats</name>
      <synopsis>Port stats</synopsis>
      <struct>
        <component componentID="1">
          <name>rx_packets</name>
          <synopsis>Total packets received</synopsis>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="2">
          <name>tx_packets</name>
          <synopsis>Total packets transmitted</synopsis>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="3">
          <name>rx_bytes</name>
          <synopsis>Total bytes received</synopsis>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="4">
          <name>tx_bytes</name>
          <synopsis>Total bytes transmitted</synopsis>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="5">
          <name>rx_errors</name>
          <synopsis>Total packet receive errors</synopsis>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="6">
          <name>tx_errors</name>
          <synopsis>Total packet transmit errors</synopsis>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="7">
          <name>rx_dropped</name>
          <synopsis>
```

```
      Total packet received and dropped. Typically because no
      space
    </synopsis>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="8">
          <name>tx_dropped</name>
          <synopsis>
      Total packet transmit dropped Typically because no space
    </synopsis>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="9">
          <name>multicast</name>
          <synopsis>Total multicast packets received</synopsis>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="10">
          <name>tx_collisions</name>
          <synopsis>Total transmit packet collisions on the
           link</synopsis>
          <optional/>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="11">
          <name>rx_length_errors</name>
          <synopsis>rx errors because of length mismatch</synopsis>
          <optional/>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="12">
          <name>rx_over_errors</name>
          <synopsis>rx errors because of buffer overflows</synopsis>
          <optional/>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="13">
          <name>rx_crc_errors</name>
          <synopsis>rx errors because of crc errors</synopsis>
          <optional/>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="14">
          <name>rx_frame_errors</name>
          <synopsis>rx errors because of frame alignment
           error</synopsis>
          <optional/>
          <typeRef>uint64</typeRef>
```

```
        </component>
        <component componentID="15">
          <name>rx_fifo_errors</name>
          <synopsis>rx errors because of fifo overruns</synopsis>
          <optional/>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="16">
          <name>rx_missed_errors</name>
          <synopsis>rx errors because of missed packets</synopsis>
          <optional/>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="17">
          <name>tx_aborted_errors</name>
          <synopsis>tx errors because of tx abort</synopsis>
          <optional/>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="18">
          <name>tx_carrier_errors</name>
          <synopsis>tx errors because of carrier problems</synopsis>
          <optional/>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="19">
          <name>tx_fifo_errors</name>
          <synopsis>tx errors because of fifo problems</synopsis>
          <optional/>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="20">
          <name>tx_heartbeat_errors</name>
          <synopsis>tx errors because of heartbeat problems</synopsis>
          <optional/>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="21">
          <name>tx_window_errors</name>
          <synopsis>tx errors because of windowing problems</synopsis>
          <optional/>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="22">
          <name>rx_compressed</name>
          <synopsis>Total rx compressed packets</synopsis>
          <optional/>
          <typeRef>uint64</typeRef>
```

```
        </component>
        <component componentID="23">
          <name>tx_compressed</name>
          <synopsis>Total tx compressed packets</synopsis>
          <optional/>
          <typeRef>uint64</typeRef>
        </component>
      </struct>
    </dataTypeDef>
    <dataTypeDef>
      <name>PortInfo</name>
      <synopsis>Describing the Port Details</synopsis>
      <struct>
        <component componentID="1">
          <name>name</name>
          <synopsis>The name of the port</synopsis>
          <optional/>
          <typeRef>string[16]</typeRef>
        </component>
        <component componentID="2">
          <name>ifindex</name>
          <synopsis>The ifindex of the port</synopsis>
          <typeRef>uint32</typeRef>
        </component>
        <component componentID="3">
          <name>L2Address</name>
          <synopsis>The MAC address</synopsis>
          <optional/>
          <typeRef>byte[6]</typeRef>
        </component>
        <component componentID="4">
          <name>mtu</name>
          <synopsis>The Maximum transmit unit for this port</synopsis>
          <optional/>
          <typeRef>uint32</typeRef>
        </component>
        <component componentID="5">
          <name>flags</name>
          <synopsis>
      flags for config and operational state. On the FE CE
      direction, these flags depend on flags mask to point to
      which flags to change
      </synopsis>
          <optional/>
          <typeRef>uint32</typeRef>
        </component>
        <component componentID="6">
          <name>flagsmask</name>
```

```
        <synopsis>
      Mask for flags for config and operational state In config
      direction, a bit turned on indicates that the FE is to set
      the corresponding flags to value specified
      </synopsis>
        <optional/>
        <typeRef>uint32</typeRef>
      </component>
      <component componentID="7">
        <name>stats</name>
        <synopsis>The 64-bit port stats</synopsis>
        <optional/>
        <typeRef>devstats</typeRef>
      </component>
      <component componentID="8">
        <name>operstate</name>
        <synopsis>The Link operational state of the port</synopsis>
        <optional/>
        <typeRef>operstates</typeRef>
      </component>
      <component componentID="9">
        <name>operstate</name>
        <synopsis>The Link operational state of the port</synopsis>
        <optional/>
        <typeRef>adminstates</typeRef>
      </component>
      <component componentID="10">
        <name>promiscuity</name>
        <synopsis>The port promiscuity</synopsis>
        <optional/>
        <typeRef>uint32</typeRef>
      </component>
    </struct>
  </dataTypeDef>
  <!-- For Classifier -->
  <dataTypeDef>
    <name>ethaddrs_key</name>
    <synopsis>Ethernet address key structure</synopsis>
    <struct>
      <component componentID="1">
        <name>eth_dst</name>
        <synopsis>Destination Ethernet address</synopsis>
        <optional/>
        <typeRef>byte[6]</typeRef>
      </component>
      <component componentID="2">
        <name>eth_dst_mask</name>
        <synopsis>Destination Ethernet address mask</synopsis>
```

```
          <optional/>
          <typeRef>byte[6]</typeRef>
        </component>
        <component componentID="3">
          <name>eth_src</name>
          <synopsis>Source Ethernet address</synopsis>
          <optional/>
          <typeRef>byte[6]</typeRef>
        </component>
        <component componentID="4">
          <name>eth_src_mask</name>
          <synopsis>Source Ethernet address mask</synopsis>
          <optional/>
          <typeRef>byte[6]</typeRef>
        </component>
      </struct>
    </dataTypeDef>
    <dataTypeDef>
      <name>inaddr_key</name>
      <synopsis>IPv4 Address key structure</synopsis>
      <struct>
        <component componentID="1">
          <name>src</name>
          <synopsis>IP Source address (BE)</synopsis>
          <optional/>
          <typeRef>octetstring[4]</typeRef>
        </component>
        <component componentID="2">
          <name>src_mask</name>
          <synopsis>IP Source address mask (BE)</synopsis>
          <optional/>
          <typeRef>octetstring[4]</typeRef>
        </component>
        <component componentID="3">
          <name>dst</name>
          <synopsis>IP Destination address (BE)</synopsis>
          <optional/>
          <typeRef>octetstring[4]</typeRef>
        </component>
        <component componentID="4">
          <name>dst_mask</name>
          <synopsis>IP Destination address mask (BE)</synopsis>
          <optional/>
          <typeRef>octetstring[4]</typeRef>
        </component>
      </struct>
    </dataTypeDef>
    <dataTypeDef>
```

```
      <name>PPPoE_control_key</name>
      <synopsis>PPPoE control key structure</synopsis>
      <struct>
        <component componentID="1">
          <name>PPPoEControl</name>
          <synopsis>PPPoE Control Traffic. Default 0x8863</synopsis>
          <optional/>
          <typeRef>uint16</typeRef>
        </component>
        <component componentID="2">
          <name>PPPControl</name>
          <synopsis>PPP Control Protocol Traffic</synopsis>
          <optional/>
          <typeRef>uint16</typeRef>
        </component>
      </struct>
    </dataTypeDef>
    <dataTypeDef>
      <name>PPPoE_subscriber_key</name>
      <synopsis>PPPoE control key structure</synopsis>
      <struct>
        <component componentID="1">
          <name>PPPSessionID</name>
          <synopsis>Session ID</synopsis>
          <optional/>
          <typeRef>uint16</typeRef>
        </component>
        <component componentID="2">
          <name>SubMACAddress</name>
          <synopsis>Session ID</synopsis>
          <optional/>
          <typeRef>octetstring[6]</typeRef>
        </component>
      </struct>
    </dataTypeDef>
    <dataTypeDef>
      <name>keyinfo</name>
      <synopsis>Describes the BNG classifier key</synopsis>
      <struct>
        <component componentID="1">
          <name>ip_proto</name>
          <synopsis>Transport protocols: TCP, UDP, SCTP, ICMP,
           ICMPV6</synopsis>
          <optional/>
          <atomic>
            <baseType>uchar</baseType>
            <specialValues>
              <specialValue value="1">
```

```
              <name>IPPROTO_ICMP</name>
              <synopsis/>
            </specialValue>
            <specialValue value="6">
              <name>IPPROTO_TCP</name>
              <synopsis/>
            </specialValue>
            <specialValue value="17">
              <name>IPPROTO_UDP</name>
              <synopsis/>
            </specialValue>
            <specialValue value="132">
              <name>IPPROTO_SCTP</name>
              <synopsis/>
            </specialValue>
            <specialValue value="58">
              <name>IPPROTO_ICMPV6</name>
              <synopsis/>
            </specialValue>
          </specialValues>
        </atomic>
      </component>
      <component componentID="2">
        <name>eth</name>
        <synopsis>Ethernet header key</synopsis>
        <optional/>
        <typeRef>ethaddrs_key</typeRef>
      </component>
      <component componentID="3">
        <name>ip</name>
        <synopsis>IP header key</synopsis>
        <optional/>
        <typeRef>inaddr_key</typeRef>
      </component>
      <component componentID="4">
        <name>PPPControl</name>
        <synopsis>PPPoE control headers</synopsis>
        <optional/>
        <typeRef>PPPoE_control_key</typeRef>
      </component>
      <component componentID="5">
        <name>PPPSubscriberTraffic_key</name>
        <synopsis>PPPoE data headers</synopsis>
        <optional/>
        <typeRef>PPPoE_subscriber_key</typeRef>
      </component>
    </struct>
  </dataTypeDef>
```

```
    <dataTypeDef>
      <name>filteractinfo</name>
      <synopsis>Basic filter action row entry</synopsis>
      <struct>
        <component componentID="1">
          <name>factype</name>
          <synopsis>Action type name</synopsis>
          <typeRef>string[16]</typeRef>
        </component>
        <component componentID="2">
          <name>faction</name>
          <synopsis>Action</synopsis>
          <atomic>
            <baseType>uchar</baseType>
            <specialValues>
              <specialValue value="0">
                <name>Drop</name>
                <synopsis>Drop packet</synopsis>
              </specialValue>
              <specialValue value="1">
                <name>ForwardPacket</name>
                <synopsis>Forward packet</synopsis>
              </specialValue>
              <specialValue value="2">
                <name>Redirect</name>
                <synopsis>Redirect packet to controller</synopsis>
              </specialValue>
            </specialValues>
          </atomic>
        </component>
      </struct>
    </dataTypeDef>
    <dataTypeDef>
      <name>ClassifierFilterInfo</name>
      <synopsis>Basic filter row entry</synopsis>
      <struct>
        <component componentID="1">
          <name>fname</name>
          <synopsis>Filter type name</synopsis>
          <typeRef>string[16]</typeRef>
        </component>
        <component componentID="2">
          <name>actions</name>
          <synopsis>The actions graph</synopsis>
          <array type="variable-size">
            <typeRef>filteractinfo</typeRef>
          </array>
        </component>
```

```
        <component componentID="3">
          <name>keys</name>
          <synopsis>Match filter keys</synopsis>
          <optional/>
          <array type="variable-size">
            <typeRef>keyinfo</typeRef>
          </array>
        </component>
      </struct>
    </dataTypeDef>
    <!-- For PPPoE LFB -->
    <dataTypeDef>
      <name>PPPoETableEntry</name>
      <synopsis>PPPoE Table Entry</synopsis>
      <struct>
        <component componentID="1">
          <name>SubscriberID</name>
          <synopsis>Subscriber Identifier</synopsis>
          <typeRef>uint32</typeRef>
        </component>
        <component componentID="2">
          <name>PPPSessionID</name>
          <synopsis>Session ID</synopsis>
          <typeRef>uint16</typeRef>
        </component>
        <component componentID="3">
          <name>SubscriberMACAddress</name>
          <synopsis>MAC Address</synopsis>
          <typeRef>octetstring[6]</typeRef>
        </component>
        <component componentID="4">
          <name>LocalMACAddress</name>
          <synopsis>Local MAC Address</synopsis>
          <typeRef>octetstring[6]</typeRef>
        </component>
        <component componentID="5">
          <name>MSS</name>
          <synopsis>Maximum Segment Size</synopsis>
          <optional/>
          <typeRef>uint16</typeRef>
        </component>
        <component componentID="6">
          <name>MRU</name>
          <synopsis>Maximum Receive Unit</synopsis>
          <optional/>
          <typeRef>uint16</typeRef>
        </component>
        <component componentID="7">
```

```
          <name>MagicNumber</name>
          <synopsis>PPP magic number</synopsis>
          <optional/>
          <typeRef>uint32</typeRef>
        </component>
        <component componentID="8">
          <name>PeerMagicNumber</name>
          <synopsis>Peer PPP magic number</synopsis>
          <optional/>
          <typeRef>uint32</typeRef>
        </component>
      </struct>
    </dataTypeDef>
    <dataTypeDef>
      <name>SubcriberStats</name>
      <synopsis>Subscriber Statistics</synopsis>
      <struct>
        <component componentID="1">
          <name>SubscriberID</name>
          <synopsis>ID of the subscriber</synopsis>
          <typeRef>uint32</typeRef>
        </component>
        <component componentID="2">
          <name>rx_encap_packets</name>
          <synopsis>Total packets received at the encap side</synopsis>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="3">
          <name>tx_encap_packets</name>
          <synopsis>Total packets transmitted at the encap side</synopsis>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="4">
          <name>rx_encap_bytes</name>
          <synopsis>Total bytes received at the encap side</synopsis>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="5">
          <name>tx_encap_bytes</name>
          <synopsis>Total bytes transmitted at the encap side</synopsis>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="6">
          <name>rx_decap_packets</name>
          <synopsis>Total packets received at the encap side</synopsis>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="7">
```

```
          <name>tx_decap_packets</name>
          <synopsis>Total packets transmitted at the encap side</synopsis>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="8">
          <name>rx_decap_bytes</name>
          <synopsis>Total bytes received at the encap side</synopsis>
          <typeRef>uint64</typeRef>
        </component>
        <component componentID="9">
          <name>tx_decap_bytes</name>
          <synopsis>Total bytes transmitted at the encap side</synopsis>
          <typeRef>uint64</typeRef>
        </component>
      </struct>
    </dataTypeDef>
    <!-- For IPv4 Routing LFB -->
    <dataTypeDef>
      <name>SubscriberRoutingTableEntry</name>
      <synopsis>A routing table entry</synopsis>
      <struct>
        <component componentID="1">
          <name>SubscriberID</name>
          <synopsis>Subscriber Identifier. Has been generated
           upstream</synopsis>
          <optional/>
          <typeRef>uint32</typeRef>
        </component>
        <component componentID="2">
          <name>IPv4Address</name>
          <synopsis>The destination IPv4 address</synopsis>
          <typeRef>octetstring[4]</typeRef>
        </component>
        <component componentID="3">
          <name>Prefixlen</name>
          <synopsis>The prefix length</synopsis>
          <atomic>
            <baseType>uchar</baseType>
            <rangeRestriction>
              <allowedRange min="0" max="32"/>
            </rangeRestriction>
          </atomic>
        </component>
        <component componentID="4">
          <name>HopSelector</name>
          <synopsis>
                 The HopSelector produced by the prefix matching LFB,
                 which will be used as an array index to find next-hop
```

```
                information.</synopsis>
          <typeRef>uint32</typeRef>
        </component>
      </struct>
    </dataTypeDef>
    <dataTypeDef>
      <name>IPv4NextHopInfoType</name>
      <synopsis>
           Data type for entry of IPv4 next-hop information table
           in IPv4NextHop LFB.  The table uses a hop selector
           received from upstream LFB as a search key to look up
           index of the table to find the next-hop information.
         </synopsis>
      <struct>
        <component componentID="1">
          <name>OutIfiIndex</name>
          <synopsis>
                The interface index of the port that is to pass
                onto downstream LFB, indicating what port this packet
                should be sent out from.</synopsis>
          <typeRef>uint32</typeRef>
        </component>
        <component componentID="2">
          <name>MTU</name>
          <synopsis>
                Maximum Transmission Unit for outgoing port
               </synopsis>
          <typeRef>uint32</typeRef>
        </component>
        <component componentID="3">
          <name>NextHopIPAddr</name>
          <synopsis>The next-hop IPv4 address</synopsis>
          <typeRef>octetstring[4]</typeRef>
        </component>
        <component componentID="4">
          <name>NextHopMACAddr</name>
          <synopsis>The next-hop MAC address</synopsis>
          <typeRef>octetstring[6]</typeRef>
        </component>
      </struct>
    </dataTypeDef>
    <!-- For PolicerLFB -->
    <dataTypeDef>
      <name>PolicerTableEntry</name>
      <synopsis>A routing table entry</synopsis>
      <struct>
        <component componentID="1">
          <name>SubscriberID</name>
```

```
          <synopsis>Subscriber Identifier. Has been generated
           upstream</synopsis>
          <typeRef>uint32</typeRef>
        </component>
        <component componentID="2">
          <name>CIR</name>
          <synopsis>Committed Information Rate</synopsis>
          <typeRef>uint32</typeRef>
        </component>
        <component componentID="3">
          <name>PIR</name>
          <synopsis>Peak Information Rate </synopsis>
          <typeRef>uint32</typeRef>
        </component>
        <component componentID="4">
          <name>CBS</name>
          <synopsis>Committed Burst Size</synopsis>
          <typeRef>uint32</typeRef>
        </component>
        <component componentID="5">
          <name>PBS</name>
          <synopsis>Peak Burst Size</synopsis>
          <typeRef>uint32</typeRef>
        </component>
      </struct>
    </dataTypeDef>
  </dataTypeDefs>
  <metadataDefs>
    <metadataDef>
      <name>SubID</name>
      <synopsis>The ID of the subscriber</synopsis>
      <metadataID>1001</metadataID>
      <typeRef>uint32</typeRef>
    </metadataDef>
    <metadataDef>
      <name>OutIfIndex</name>
      <synopsis>Interface Index to output packets</synopsis>
      <metadataID>1002</metadataID>
      <typeRef>uint32</typeRef>
    </metadataDef>
  </metadataDefs>
  <LFBClassDefs>
    <LFBClassDef LFBClassID="2001">
      <name>Port</name>
      <synopsis>A Port LFB</synopsis>
      <version>1.0</version>
      <inputPorts>
        <inputPort>
```

```
          <name>IngressInPort</name>
          <synopsis>Ingress port from outside the BNG to be
           sent inside</synopsis>
          <expectation>
            <frameExpected>
              <ref>Arbitraty</ref>
            </frameExpected>
          </expectation>
        </inputPort>
        <inputPort>
          <name>EgressInPort</name>
          <synopsis>Egress port from within the BNG to be
           sent outside</synopsis>
          <expectation>
            <frameExpected>
              <ref>Arbitraty</ref>
            </frameExpected>
            <metadataExpected>
              <ref>OutIfIndex</ref>
            </metadataExpected>
          </expectation>
        </inputPort>
      </inputPorts>
      <outputPorts>
        <outputPort>
          <name>IngressOutPort</name>
          <synopsis>Ingress port to send packets within the
           BNG</synopsis>
          <product>
            <frameProduced>
              <ref>Arbitrary</ref>
            </frameProduced>
          </product>
        </outputPort>
        <outputPort>
          <name>EgressOutPort</name>
          <synopsis>Egress port to send packets out from the
           BNG</synopsis>
          <product>
            <frameProduced>
              <ref>Arbitrary</ref>
            </frameProduced>
          </product>
        </outputPort>
      </outputPorts>
      <components>
        <component componentID="1" access="read-write">
          <name>ports</name>
```

```
          <synopsis>the table of all ports</synopsis>
          <array type="variable-size">
            <typeRef>PortInfo</typeRef>
          </array>
        </component>
      </components>
      <events baseID="161">
        <event eventID="1">
          <name>PortChanged</name>
          <synopsis>
           An existing port has been updated.
           When the change occurs we report the table row that has
           changed including its contents + index (port ifindex).
          </synopsis>
          <eventTarget>
            <eventField>ports</eventField>
          </eventTarget>
          <eventChanged/>
          <eventReports>
            <eventReport>
              <eventField>ports</eventField>
              <eventSubscript>_pifindex_</eventSubscript>
            </eventReport>
          </eventReports>
        </event>
        <event eventID="2">
          <name>PortDeleted</name>
          <synopsis>
           An existing port has been deleted.
           When the change occurs we report the table row that
           has changed including its contents + index (port ifindex).
          </synopsis>
          <eventTarget>
            <eventField>ports</eventField>
          </eventTarget>
          <eventDeleted/>
          <eventReports>
            <eventReport>
              <eventField>ports</eventField>
              <eventSubscript>_pifindex_</eventSubscript>
            </eventReport>
          </eventReports>
        </event>
        <event eventID="3">
          <name>PortCreated</name>
          <synopsis>
        A new port has been created. When the change occurs we
        report the table row that has changed including its
```

```
        contents + index (port ifindex).
        </synopsis>
          <eventTarget>
            <eventField>ports</eventField>
          </eventTarget>
          <eventCreated/>
          <eventReports>
            <eventReport>
              <eventField>ports</eventField>
              <eventSubscript>_pifindex_</eventSubscript>
            </eventReport>
          </eventReports>
        </event>
        <event eventID="4">
          <name>PortStatsChanged</name>
          <synopsis>
        Event used to advertise synchronously port stats. The
        ForCES eventInterval property is useful for specifying the
        synchronous interval.
        </synopsis>
          <eventTarget>
            <eventField>ports</eventField>
          </eventTarget>
          <eventChanged/>
          <eventReports>
            <eventReport>
              <eventField>ports</eventField>
              <eventSubscript>_pifindex_</eventSubscript>
            </eventReport>
          </eventReports>
        </event>
      </events>
    </LFBClassDef>
    <LFBClassDef LFBClassID="2002">
      <name>Classifier</name>
      <synopsis>A Classifier LFB. Classifies frames</synopsis>
      <version>1.0</version>
      <inputPorts>
        <inputPort>
          <name>InPort</name>
          <synopsis>Input for the Classifier. Input could be from
          Port or tunneling infrastructure</synopsis>
          <expectation>
            <frameExpected>
              <ref>Arbitrary</ref>
            </frameExpected>
          </expectation>
        </inputPort>
```

```
      </inputPorts>
      <outputPorts>
        <outputPort>
          <name>ControlOut</name>
          <synopsis>Redirects packet towards the control plane.
          </synopsis>
          <product>
            <frameProduced>
              <ref>Arbitrary</ref>
            </frameProduced>
          </product>
        </outputPort>
        <outputPort>
          <name>EthernetOut</name>
          <synopsis>Port to send Ethenet frames</synopsis>
          <product>
            <frameProduced>
              <ref>EthernetFrame</ref>
            </frameProduced>
            <metadataProduced>
              <ref>SubID</ref>
            </metadataProduced>
          </product>
        </outputPort>
        <outputPort>
          <name>IPv4Out</name>
          <synopsis>Port to send IPv4 packets</synopsis>
          <product>
            <frameProduced>
              <ref>IPv4Packet</ref>
            </frameProduced>
            <metadataProduced>
              <ref>SubID</ref>
            </metadataProduced>
          </product>
        </outputPort>
      </outputPorts>
      <components>
        <component componentID="1" access="read-write">
          <name>Filters</name>
          <synopsis>The table of filters</synopsis>
          <array type="variable-size">
            <typeRef>ClassifierFilterInfo</typeRef>
          </array>
        </component>
      </components>
      <events baseID="61">
        <event eventID="1">
```

```
      <name>FilterChanged</name>
      <synopsis>
         A Filter instance has been updated. When the change occurs
         we report the table row that has changed including
         its contents + index.
      </synopsis>
      <eventTarget>
        <eventField>Filters</eventField>
      </eventTarget>
      <eventChanged/>
      <eventReports>
        <eventReport>
          <eventField>Filters</eventField>
          <eventSubscript>_findex_</eventSubscript>
        </eventReport>
      </eventReports>
    </event>
    <event eventID="2">
      <name>FilterDeleted</name>
      <synopsis>
         An existing Filter has been deleted. When the change
         occurs we report the table row that has changed including
         its contents + index.
      </synopsis>
      <eventTarget>
        <eventField>Filters</eventField>
      </eventTarget>
      <eventDeleted/>
      <eventReports>
        <eventReport>
          <eventField>Filters</eventField>
          <eventSubscript>_findex_</eventSubscript>
        </eventReport>
      </eventReports>
    </event>
    <event eventID="3">
      <name>FilterCreated</name>
      <synopsis>
          A new filter has been created. When the change occurs
          we report the table row that has changed including
          its contents + index.
       </synopsis>
      <eventTarget>
        <eventField>Filters</eventField>
      </eventTarget>
      <eventCreated/>
      <eventReports>
        <eventReport>
```

```
            <eventField>Filters</eventField>
            <eventSubscript>_findex_</eventSubscript>
          </eventReport>
        </eventReports>
      </event>
    </events>
  </LFBClassDef>
  <LFBClassDef LFBClassID="2003">
    <name>PPPoE</name>
    <synopsis>PPPoE LFB to encap and decap packets.</synopsis>
    <version>1.0</version>
    <inputPorts>
      <inputPort>
        <name>EncapIn</name>
        <synopsis>A port to encapsulate an IP packet</synopsis>
        <expectation>
          <frameExpected>
            <ref>IPv4Packet</ref>
          </frameExpected>
          <metadataExpected>
            <ref>SubID</ref>
            <ref>OutIfIndex</ref>
          </metadataExpected>
        </expectation>
      </inputPort>
      <inputPort>
        <name>DecapIn</name>
        <synopsis>A port to decapsulate a PPPoE packet</synopsis>
        <expectation>
          <frameExpected>
            <ref>EthernetFrame</ref>
          </frameExpected>
          <metadataExpected>
            <ref>SubID</ref>
          </metadataExpected>
        </expectation>
      </inputPort>
    </inputPorts>
    <outputPorts>
      <outputPort>
        <name>EncapOut</name>
        <synopsis>After Encaping an IPv4 Packet create an Ethernet
         frame</synopsis>
        <product>
          <frameProduced>
            <ref>EthernetFrame</ref>
          </frameProduced>
        </product>
```

```
        </outputPort>
        <outputPort>
          <name>DecapOut</name>
          <synopsis>Generates IPv4 packets</synopsis>
          <product>
            <frameProduced>
              <ref>IPv4Packet</ref>
            </frameProduced>
            <metadataProduced>
              <ref>SubID</ref>
            </metadataProduced>
          </product>
        </outputPort>
      </outputPorts>
      <components>
        <component componentID="1" access="read-write">
          <name>PPPoEInfo</name>
          <synopsis>Table with PPPoE Subscriber Information</synopsis>
          <array>
            <typeRef>PPPoETableEntry</typeRef>
          </array>
        </component>
        <component componentID="2" access="read-only">
          <name>Stats</name>
          <synopsis>Table with statistics for Encap and Decap
           packets per subscriber</synopsis>
          <array>
            <typeRef>SubcriberStats</typeRef>
          </array>
        </component>
      </components>
      <events baseID="161">
        <event eventID="1">
          <name>SubChanged</name>
          <synopsis>
           An existing PPPoE Subscriber has been updated.
           When the change occurs we report the table row that has
           changed including its contents + subscriberID.
        </synopsis>
          <eventTarget>
            <eventField>PPPoEInfo</eventField>
          </eventTarget>
          <eventChanged/>
          <eventReports>
            <eventReport>
              <eventField>PPPoEInfo</eventField>
              <eventSubscript>_SubscriberID_</eventSubscript>
            </eventReport>
```

```
      </eventReports>
    </event>
    <event eventID="2">
      <name>SubDeleted</name>
      <synopsis>
     An existing PPPoE Subscriber has been deleted.
     When the change occurs we report the table row that has
     changed including its contents + subscriberID.
    </synopsis>
      <eventTarget>
        <eventField>PPPoEInfo</eventField>
      </eventTarget>
      <eventDeleted/>
      <eventReports>
        <eventReport>
          <eventField>PPPoEInfo</eventField>
          <eventSubscript>_SubscriberID_</eventSubscript>
        </eventReport>
      </eventReports>
    </event>
    <event eventID="3">
      <name>SubCreated</name>
      <synopsis>
 A new PPPoE Subscriber has been created.
When the change occurs we report the table row that has
changed including its contents + subscriberID.
 </synopsis>
      <eventTarget>
        <eventField>PPPoEInfo</eventField>
      </eventTarget>
      <eventCreated/>
      <eventReports>
        <eventReport>
          <eventField>PPPoEInfo</eventField>
          <eventSubscript>_SubscriberID_</eventSubscript>
        </eventReport>
      </eventReports>
    </event>
    <event eventID="4">
      <name>StatsChanged</name>
      <synopsis>
 Event used to advertise synchronously encap stats. The
 ForCES eventInterval property is useful for specifying the
 synchronous interval.
 </synopsis>
      <eventTarget>
        <eventField>Stats</eventField>
      </eventTarget>
```

```
          <eventChanged/>
          <eventReports>
            <eventReport>
              <eventField>Stats</eventField>
              <eventSubscript>_SubscriberID_</eventSubscript>
            </eventReport>
          </eventReports>
        </event>
      </events>
    </LFBClassDef>
    <LFBClassDef LFBClassID="2004">
      <name>IPv4Routing</name>
      <synopsis>IPv4 Routing LFB</synopsis>
      <version>1.0</version>
      <inputPorts>
        <inputPort>
          <name>InPort</name>
          <synopsis>Input port for packets</synopsis>
          <expectation>
            <frameExpected>
              <ref>IPv4Packet</ref>
            </frameExpected>
            <metadataExpected>
              <ref>SubID</ref>
            </metadataExpected>
          </expectation>
        </inputPort>
      </inputPorts>
      <outputPorts>
        <outputPort>
          <name>NormalOut</name>
          <synopsis>Output port for packets</synopsis>
          <product>
            <frameProduced>
              <ref>IPv4Packet</ref>
            </frameProduced>
            <metadataProduced>
              <ref>SubID</ref>
              <ref>OutIfIndex</ref>
            </metadataProduced>
          </product>
        </outputPort>
        <outputPort>
          <name>ExceptionOut</name>
          <synopsis>Port for errors</synopsis>
          <product>
            <frameProduced>
              <ref>IPv4Packet</ref>
```

```
          </frameProduced>
        </product>
      </outputPort>
    </outputPorts>
    <components>
      <component componentID="1" access="read-write">
        <name>IPv4RoutingTable</name>
        <synopsis>
         A table for IPv4 Longest Prefix Match.  The
         destination IPv4 address of every input packet is
         used as a search key to look up the table to find
         out a next-hop selector.
         </synopsis>
        <array>
          <typeRef>SubscriberRoutingTableEntry</typeRef>
        </array>
      </component>
      <component componentID="2" access="read-write">
        <name>IPv4NextHopTable</name>
        <synopsis>
         The IPv4NextHopTable component.  A
         HopSelector is used to match the table index
         to find out a row that contains the next-hop
         information result.
         </synopsis>
        <array>
          <typeRef>IPv4NextHopInfoType</typeRef>
        </array>
      </component>
    </components>
  </LFBClassDef>
  <LFBClassDef LFBClassID="2005">
    <name>Policer</name>
    <synopsis>Policer LFB</synopsis>
    <version>1.0</version>
    <inputPorts>
      <inputPort>
        <name>InUpstreamPort</name>
        <synopsis>Input port for the Policer LFB for upstream
         traffic</synopsis>
        <expectation>
          <frameExpected>
            <ref>IPv4Packet</ref>
          </frameExpected>
          <metadataExpected>
            <ref>SubID</ref>
          </metadataExpected>
        </expectation>
```

```
        </inputPort>
        <inputPort>
          <name>InDownstreamPort</name>
          <synopsis>Input port for the Policer LFB  for downstream
           traffic</synopsis>
          <expectation>
            <frameExpected>
              <ref>IPv4Packet</ref>
            </frameExpected>
            <metadataExpected>
              <ref>SubID</ref>
            </metadataExpected>
          </expectation>
        </inputPort>
      </inputPorts>
      <outputPorts>
        <outputPort>
          <name>OutUpstreamPort</name>
          <synopsis>Output port for the Policer LFB  for upstream
           traffic</synopsis>
          <product>
            <frameProduced>
              <ref>IPv4Packet</ref>
            </frameProduced>
            <metadataProduced>
              <ref>SubID</ref>
            </metadataProduced>
          </product>
        </outputPort>
        <outputPort>
          <name>OutDownstreamPort</name>
          <synopsis>Output port for the Policer LFB  for downstream
           traffic</synopsis>
          <product>
            <frameProduced>
              <ref>IPv4Packet</ref>
            </frameProduced>
            <metadataProduced>
              <ref>SubID</ref>
            </metadataProduced>
          </product>
        </outputPort>
      </outputPorts>
      <components>
        <component componentID="1" access="read-write">
          <name>UpstreamPolicy</name>
          <synopsis>Policy entries for upstream traffic</synopsis>
          <array>
```

```
            <typeRef>BandwidthTableEntry</typeRef>
          </array>
        </component>
        <component componentID="2" access="read-write">
          <name>DownstreamPolicy</name>
          <synopsis>Policy entries for downstream traffic</synopsis>
          <array>
            <typeRef>PolicerTableEntry</typeRef>
          </array>
        </component>
      </components>
    </LFBClassDef>
  </LFBClassDefs>
</LFBLibrary>
```

Figure 9: BNG XML model

8. Acknowledgements

The activities of Evangelos Haleplidis have been carried out with funding provided via the StandICT.eu initiative, funded with Grant Agreement no. 780439 under the European Commission's Horizon 2020 Programme.

9. IANA Considerations

TBD

10. Security Considerations

TBD

11. References

11.1. Normative References

[I-D.haleplidis-bcause-forces-gap-analysis]
Haleplidis, E. and J. Salim, "ForCES architecture CUSP applicability", draft-haleplidis-bcause-forces-gap-analysis-00 (work in progress), April 2019.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-23 (work in progress), September 2019.

[RFC2516] Mamakos, L., Lidl, K., Evarts, J., Carrel, D., Simone, D., and R. Wheeler, "A Method for Transmitting PPP Over Ethernet (PPPoE)", RFC 2516, DOI 10.17487/RFC2516, February 1999, <https://www.rfc-editor.org/info/rfc2516>.

[RFC3746] Yang, L., Dantu, R., Anderson, T., and R. Gopal, "Forwarding and Control Element Separation (ForCES) Framework", RFC 3746, DOI 10.17487/RFC3746, April 2004, <https://www.rfc-editor.org/info/rfc3746>.

[RFC5810] Doria, A., Ed., Hadi Salim, J., Ed., Haas, R., Ed., Khosravi, H., Ed., Wang, W., Ed., Dong, L., Gopal, R., and J. Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification", RFC 5810, DOI 10.17487/RFC5810, March 2010, <https://www.rfc-editor.org/info/rfc5810>.

[RFC5811] Hadi Salim, J. and K. Ogawa, "SCTP-Based Transport Mapping Layer (TML) for the Forwarding and Control Element Separation (ForCES) Protocol", RFC 5811, DOI 10.17487/RFC5811, March 2010, <https://www.rfc-editor.org/info/rfc5811>.

[RFC5812] Halpern, J. and J. Hadi Salim, "Forwarding and Control Element Separation (ForCES) Forwarding Element Model", RFC 5812, DOI 10.17487/RFC5812, March 2010, <https://www.rfc-editor.org/info/rfc5812>.

[RFC7121] Ogawa, K., Wang, W., Haleplidis, E., and J. Hadi Salim, "High Availability within a Forwarding and Control Element Separation (ForCES) Network Element", RFC 7121, DOI 10.17487/RFC7121, February 2014, <https://www.rfc-editor.org/info/rfc7121>.

[TR-101] Broadband Forum, "TR-101 - Migration to Ethernet-Based Broadband Aggregation", 2011, <https://www.broadband-forum.org/download/TR-101_Issue-2.pdf>.

11.2. Informative References

[media1] "Forces-vzn", , 06 2016, <https://www.sdxcentral.com/articles/news/verizon-uses-radisys-mojatatu-sdn-nfv/2016/06/>.

[media2] "Forces-vzn2", , 06 2016, <https://www.sdxcentral.com/articles/news/meet-mojatatu-quiet-company-verizon-chose-sdn/2016/06/>.

[media3] "Forces-Large-Scale-Kubernetes", , 10 2019, <https://www.cengn.ca/mojatatu-analysis-kubernetes-deployments/>.

[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>.

[RFC6956] Wang, W., Haleplidis, E., Ogawa, K., Li, C., and J. Halpern, "Forwarding and Control Element Separation (ForCES) Logical Function Block (LFB) Library", RFC 6956, DOI 10.17487/RFC6956, June 2013, <https://www.rfc-editor.org/info/rfc6956>.

Authors' Addresses

Evangelos Haleplidis
M. Aleksandrou 29B
Paiania, Athens 19002
Greece

Email: ehalep@gmail.com

Jamal Hadi Salim
Mojatatu Networks
Suite 200, 15 Fitzgerald Road
Ottawa, Ontario K2H 9G1
Canada

Email: hadi@mojatatu.com

Jae Won Chung
Viasat
6155 El Camino Real
Carlsbad 92009
USA

Email: JaeWon.Chung@viasat.com