Internet Engineering Task Force   W. Wang
Internet-Draft   Zhejiang Gongshang University
Intended status: Standards Track   E. Haleplidis
Expires: September 16, 2013   University of Patras
                                                       K. Ogawa
                                                NTT Corporation
                                                         C. Li
                                               Hangzhou DPtech
                                                    J. Halpern
                                                      Ericsson
                                               March 15, 2013

ForCES Logical Function Block (LFB) Library
draft-ietf-forces-lfb-lib-11

Abstract

   This document defines basic classes of Logical Function Blocks (LFBs)
   used in the Forwarding and Control Element Separation (ForCES). The
   basic LFB classes are defined according to ForCES FE model and ForCES
   protocol specifications, and are scoped to meet requirements of
   typical router functions and considered as the basic LFB library for
   ForCES. The library includes the descriptions of the LFBs and the
   XML definitions.

Status of this Memo

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

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

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

   This Internet-Draft will expire on September 16, 2013.

Copyright Notice

Table of Contents

1. Terminology and Conventions

1.1. 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. Definitions

This document follows the terminology defined by the ForCES protocol
in [RFC5810] and by the ForCES FE model in [RFC5812]. The
definitions below are repeated for clarity.

   Control Element (CE) - A logical entity that implements the ForCES
   protocol and uses it to instruct one or more FEs on how to process
   packets. CEs handle functionality such as the execution of
   control and signaling protocols.

   Forwarding Element (FE) - A logical entity that implements the
   ForCES protocol. FEs use the underlying hardware to provide per-
   packet processing and handling as directed/controlled by one or
   more CEs via the ForCES protocol.

   ForCES Network Element (NE) - An entity composed of one or more
   CEs and one or more FEs. To entities outside an NE, the NE
   represents a single point of management. Similarly, an NE usually
   hides its internal organization from external entities.

   LFB (Logical Function Block) - The basic building block that is
   operated on by the ForCES protocol. The LFB is a well defined,
   logically separable functional block that resides in an FE and is
   controlled by the CE via ForCES protocol. The LFB may reside at
   the FE's datapath and process packets or may be purely an FE
   control or configuration entity that is operated on by the CE.
   Note that the LFB is a functionally accurate abstraction of the
   FE's processing capabilities, but not a hardware-accurate
   representation of the FE implementation.

   FE Model - The FE model is designed to model the logical
   processing functions of an FE, which is defined by the ForCES FE
   model document [RFC5812]. The FE model proposed in this document
   includes three components; the LFB modeling of individual Logical
   Functional Block (LFB model), the logical interconnection between
   LFBs (LFB topology), and the FE-level attributes, including FE
   capabilities. The FE model provides the basis to define the
   information elements exchanged between the CE and the FE in the
   ForCES protocol [RFC5810].

   FE Topology - A representation of how the multiple FEs within a
   single NE are interconnected. Sometimes this is called inter-FE
   topology, to be distinguished from intra-FE topology (i.e., LFB
   topology).

LFB Class and LFB Instance - LFBs are categorized by LFB Classes.
An LFB Instance represents an LFB Class (or Type) existence.
There may be multiple instances of the same LFB Class (or Type) in
an FE. An LFB Class is represented by an LFB Class ID, and an LFB
Instance is represented by an LFB Instance ID. As a result, an
LFB Class ID associated with an LFB Instance ID uniquely specifies
an LFB existence.

LFB Metadata - Metadata is used to communicate per-packet state
from one LFB to another, but is not sent across the network. The
FE model defines how such metadata is identified, produced and
consumed by the LFBs. It defines the functionality but not how
metadata is encoded within an implementation.

LFB Component - Operational parameters of the LFBs that must be
visible to the CEs are conceptualized in the FE model as the LFB
components. The LFB components include, for example, flags,
single parameter arguments, complex arguments, and tables that the
CE can read and/or write via the ForCES protocol (see below).

LFB Topology - Representation of how the LFB instances are
logically interconnected and placed along the datapath within one
FE. Sometimes it is also called intra-FE topology, to be
distinguished from inter-FE topology.

Data Path - A conceptual path taken by packets within the
forwarding plane inside an FE. Note that more than one data path
can exist within an FE.

ForCES Protocol - While there may be multiple protocols used
within the overall ForCES architecture, the term "ForCES protocol"
and "protocol" refer to the Fp reference points in the ForCES
Framework in [RFC3746]. This protocol does not apply to CE-to-CE
communication, FE-to-FE communication, or to communication between
FE and CE managers. Basically, the ForCES protocol works in a
master-slave mode in which FEs are slaves and CEs are masters.
This document defines the specifications for this ForCES protocol.

LFB Port - A port refers to an LFB input port or output port. See
Section 3.2 of [RFC5812] for more detailed definitions.

Physical Port - A port refers to a physical media input port or
output port of an FE. A physical port is usually assigned with a
physical port ID, abbreviated with a PHYPortID. This document
mainly deals with physical ports with Ethernet media.

Logical Port - A conceptually virtual port at data link layer (L2)
or network layer (L3). A logical port is usually assigned with a
logical port ID, abbreviated with a LogicalPortID. The logical
ports can be further categorized with a L2 logical port or a L3
logical port. An L2 logical port can be assigned with a L2
logical port ID, abbreviated with a L2PortID. An L3 logical port
can be assigned with a L3 logical port ID, abbreviated with a
L3PortID. MAC layer VLAN ports belongs to L2 logical ports as
well as logical ports.

LFB Class Library - The LFB class library is a set of LFB classes
that has been identified as the most common functions found in
most FEs and hence should be defined first by the ForCES Working
Group. The LFB Class Library is defined by this document.

3. Introduction

   [RFC5810] specifies Forwarding and Control Element Separation
   (ForCES) framework. In the framework, Control Elements (CEs)
   configure and manage one or more separate Forwarding Elements (FEs)
   within a Network Element (NE) by use of a ForCES protocol. [RFC5810]
   specifies the ForCES protocol. [RFC5812] specifies the Forwarding
   Element (FE) model. In the model, resources in FEs are described by
   classes of Logical Function Blocks (LFBs). The FE model defines the
   structure and abstract semantics of LFBs, and provides XML schema for
   the definitions of LFBs.

   This document conforms to the specifications of the FE model
   [RFC5812] and specifies detailed definitions of classes of LFBs,
   including detailed XML definitions of LFBs. These LFBs form a base
   LFB library for ForCES. LFBs in the base library are expected to be
   combined to form an LFB topology for a typical router to implement IP
   forwarding. It should be emphasized that an LFB is an abstraction of
   functions rather than its implementation details. The purpose of the
   LFB definitions is to represent functions so as to provide
   interoperability between separate CEs and FEs.

   More LFB classes with more functions may be developed in future time
   and documented by IETF. Vendors may also develop proprietary LFB
   classes as described in the FE model [RFC5812].

3.1. Scope of the Library

   It is intended that the LFB classes described in this document are
   designed to provide the functions of a typical router. [RFC1812]
   specifies that a typical router is expected to provide functions to:

   (1) Interface to packet networks and implement the functions
        required by that network. These functions typically include:

        * Encapsulating and decapsulating the IP datagrams with the
           connected network framing (e.g., an Ethernet header and
           checksum),

        * Sending and receiving IP datagrams up to the maximum size
           supported by that network, this size is the network's Maximum
           Transmission Unit or MTU,

        * Translating the IP destination address into an appropriate
           network-level address for the connected network (e.g., an
           Ethernet hardware address), if needed, and

* Responding to network flow control and error indications, if any.

(2) Conform to specific Internet protocols including the Internet Protocol (IPv4 and/or IPv6), Internet Control Message Protocol (ICMP), and others as necessary.

(3) Receive and forward Internet datagrams. Important issues in this process are buffer management, congestion control, and fairness.

* Recognizes error conditions and generates ICMP error and information messages as required.

* Drops datagrams whose time-to-live fields have reached zero.

* Fragments datagrams when necessary to fit into the MTU of the next network.

(4) Choose a next hop destination for each IP datagram, based on the information in its routing database.

(5) Usually support an interior gateway protocol (IGP) to carry out distributed routing and reachability algorithms with the other routers in the same autonomous system. In addition, some routers will need to support an exterior gateway protocol (EGP) to exchange topological information with other autonomous systems. For all routers, it is essential to provide ability to manage static routing items.

(6) Provide network management and system support facilities, including loading, debugging, status reporting, statistcs query, exception reporting and control.

The classical IP router utilizing the ForCES framework constitutes a CE running some controlling IGP and/or EGP function or static route setup and FEs implementing using Logical Function Blocks (LFBs) conforming to the FE model[RFC5812] specifications. The CE, in conformance to the ForCES protocol[RFC5810] and the FE model [RFC5812] specifications, instructs the LFBs on the FE how to treat received/sent packets.

Packets in an IP router are received and transmitted on physical media typically referred to as "ports". Different physical port media will have different ways for encapsulating outgoing frames and decapsulating incoming frames. The different physical media will also have different attributes that influence its behavior and how frames get encapsulated or decapsulated. This document will only

deal with Ethernet physical media. Other future documents may deal
with other type of media. This document will also interchangeably
refer to a port to be an abstraction that constitutes a PHY and a MAC
as described by the LFBs like EtherPHYCop, EtherMACIn, and
EtherMACOut.

IP packets emanating from port LFBs are then processed by a
validation LFB before being further forwarded to the next LFB. After
the validation process the packet is passed to an LFB where IP
forwarding decision is made. In the IP Forwarding LFBs, a Longest
Prefix Match LFB is used to look up the destination information in a
packet and select a next hop index for sending the packet onward. A
next hop LFB uses the next hop index metadata to apply the proper
headers to the IP packets, and direct them to the proper egress.
Note that in the process of IP packets processing, in this document,
we are adhering to the weak-host model [RFC1122] since that is the
most usable model for a packet processing Network Element.

3.2. Overview of LFB Classes in the Library

It is critical to classify functional requirements into various
classes of LFBs and construct a typical but also flexible enough base
LFB library for various IP forwarding equipments.

3.2.1. LFB Design Choices

A few design principles were factored into choosing how the base LFBs
looked like. These are:

o If a function can be designed by either one LFB or two or more
   LFBs with the same cost, the choice is to go with two or more LFBs
   so as to provide more flexibility for implementers.

o An LFB should take advantage of its independence as much as
   possible and have minimal coupling with other LFBs. The coupling
   may be from LFB attributes definitions as well as physical
   implementations.

o Unless there is a clear difference in functionality, similar
   packet processing in the base LFB library should not be
   represented simultaneously as two or more LFBs. For instance, it
   should not be simultaneously defined with two different LFBs for
   the same next hop processing. Or else, it may add extra burden on
   implementation to achieve interoperability.

3.2.2. LFB Class Groupings

The document defines groups of LFBs for typical router function
requirements:

(1) A group of Ethernet processing LFBs are defined to abstract the
     packet processing for Ethernet as the port media type. As the
     most popular media type with rich processing features, Ethernet
     media processing LFBs was a natural choice. Definitions for
     processing of other port media type like POS or ATM may be
     incorporated in the library in future version of the document or
     in a future separate document. The following LFBs are defined
     for Ethernet processing:

     * EtherPHYCop (Section 5.1.1)

     * EtherMACIn (Section 5.1.2)

     * EtherClassifier (Section 5.1.3)

     * EtherEncap (Section 5.1.4)

     * EtherMACOut (Section 5.1.5)

(2) A group of LFBs are defined for IP packet validation process.
     The following LFBs are defined for IP validation processing:

     * IPv4Validator (Section 5.2.1)

     * IPv6Validator (Section 5.2.2)

(3) A group of LFBs are defined to abstract IP forwarding process.
     The following LFBs are defined for IP forwarding processing:

     * IPv4UcastLPM (Section 5.3.1)

     * IPv4NextHop (Section 5.3.2)

     * IPv6UcastLPM (Section 5.3.3)

     * IPv6NextHop (Section 5.3.4)

(4) A group of LFBs are defined to abstract the process for redirect
     operation, i.e., data packet transmission between CE and FEs.
     The following LFBs are defined for redirect processing:

     * RedirectIn (Section 5.4.1)

         * RedirectOut (Section 5.4.2)

   (5) A group of LFBs are defined for abstracting some general purpose
        packet processing. These processing processes are usually
        general to many processing locations in an FE LFB topology. The
        following LFBs are defined for redirect processing:

         * BasicMetadataDispatch (Section 5.5.1)

         * GenericScheduler (Section 5.5.2)

3.2.3. Sample LFB Class Application

   Although Section 7 will present use cases for LFBs defined in this
   document, this section shows a simple sample LFB class application in
   advance so that readers can get a quick overlook of the LFB classes
   with the usage.

   Figure 1 shows a simple LFB processing path for Ethernet packets
   entered from Ethernet physical ports.

```
+-----+                 +------+
|     |EtherPHYIn       |      |              from some LFB(s) which
|     |<---------------|Ether |<---------- generate Ethernet
|     |                 |MACOut|            packets
|     |                 | LFB  |
|Ether|                 +------+
|PHY  |                 +------+
|Cop  |                 |      |
|LFB  |EtherPHYOut      | Ether|              to some LFB(s) which
|     |--------------->| MACIn|----------> may classify Ethernet
|     |                 | LFB  |            packets and do IP layer
|     |                 |      |            processing
+-----+                 +------+
```
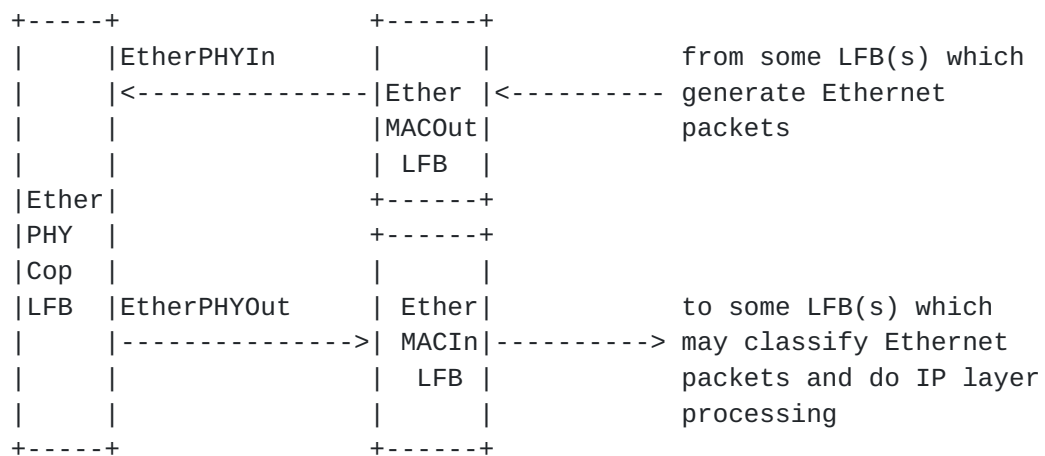
                  Figure 1: A simple sample LFB use case

   In the figure, Ethernet packets from outer networks enter via the
   EtherPHYCop LFB(Section 5.1.1), which describes Ethernet copper
   interface property(like the link speed) at physical layer. After
   physical layer process, Ethernet packets are delivered to EtherMACIn
   LFB(Section 5.1.2) to describe its MAC layer processing
   functions(like locality check). The packets after EtherMACIn LFB may
   require further processing to implement various functions(like IP
   layer forwarding),therefore some LFBs may follow the EtherMACIn LFB
   in topology to describe followed processing functions.

   Meanwhile, packets generated by some LFB(s) may need to be submitted

to outer physical networks. The process is described in the figure
by an EtherMACOut LFB(Section 5.1.5) at MAC layer and the EtherPHYCop
LFB at physical layer.

3.3. Document Structure

Base type definitions, including data types, packet frame types, and
metadata types are presented in advance for definitions of various
LFB classes. Section 4 (Base Types section) provides a description
on the base types used by this LFB library. To enable extensive use
of these base types by other LFB class definitions, the base type
definitions are provided as a separate library.

Within every group of LFB classes, a set of LFBs are defined for
individual function purposes. Section 5 (LFB Class Descriptions
section) provides text descriptions on the individual LFBs. Note
that for a complete definition of an LFB, a text description as well
as a XML definition is required.

LFB classes are finally defined by XML with specifications and schema
defined in the ForCES FE model[RFC5812]. Section 6 (XML LFB
Definitions section) provides the complete XML definitions of the
base LFB classes library.

Section 7 provides several use cases on how some typical router
functions can be implemented using the base LFB library defined in
this document.

4. Base Types

   The FE model [RFC5812] has specified predefined (built-in) atomic
   data-types as below:

   char, uchar, int16, uint16, int32, uint32, int64, uint64, string[N],
   string, byte[N], boolean, octetstring[N], float16, float32, float64.

   Note that, unlike SNMP information model SMI([RFC2578]), the FE model
   has not defined specific atomic data type for counting purpose. This
   document follows not to define specific counter types. To describe
   LFB elements for packet statistics which actually require counters on
   packets, an unsigned integer, like an uint32 or an uint64 is adopted.
   This document states that any LFB element defined for counter purpose
   is specified to monotonically increase until it reaches a maximum
   value, when it wraps around and starts increasing again from zero.
   This document also states that it is implementation's issue how the
   unsigned integer element might be maintained to cope with issues like
   counter discontinuities when a counter wraps or is reset by any
   reasons. If a CE is expected to understand more meanings of the
   counter element than stated above, a private definition on the
   element between CE and FE may be required.

   Based on the atomic data types and with the use of type definition
   elements in the FE model XML schema, new data types, packet frame
   types, and metadata types can be defined.

   To define a base LFB library for typical router functions, a set of
   base data types, frame types, and metadata types should be defined.
   This section provides a brief description of the base types and a
   full XML definition of them as well.

   The base type XML definitions are provided with a separate XML
   library file named "BaseTypeLibrary". Users can refer to this
   library by the statement:

   <load library="BaseTypeLibrary", location="..."/>

4.1. Data Types

   Data types defined in the base type library are categorized by types
   of atomic, compound struct, and compound array.

4.1.1. Atomic

   The following data types are defined as atomic data types and put in
   the base type library:

```
    Data Type Name       Brief Description
    --------------       -----------------
    IPv4Addr             IPv4 address
    IPv6Addr             IPv6 address
    IEEEMAC              IEEE MAC address
    LANSpeedType         LAN speed by value types
    DuplexType           Duplex types
    PortStatusType       The possible types of port status, used for
                          both administrative and operative status.
    VlanIDType           The type of VLAN ID
    VlanPriorityType     The type of VLAN priority
    SchdDisciplineType   Scheduling discipline type
```

[4.1.2](#). Compound Struct

The following compound struct types are defined in the base type
library:

```
  Data Type Name             Brief Description
  --------------             -----------------
  EtherDispatchEntryType     Entry type for Ethernet dispatch table
  VlanInputTableEntryType    Entry type for VLAN input table
  EncapTableEntryType        Entry type for Ethernet encapsulation table
  MACInStatsType             Statistics type for EtherMACIn LFB
  MACOutStatsType            Statistics type for EtherMACOut LFB
  EtherClassifyStatsType     Entry type for statistics table in
                              EtherClassifier LFB.
   IPv4PrefixInfoType        Entry type for IPv4 prefix table
   IPv6PrefixInfoType        Entry type for IPv6 prefix table
   IPv4NextHopInfoType       Entry type for IPv4 next hop table
   IPv6NextHopInfoType       Entry type for IPv6 next hop table
   IPv4ValidatorStatsType    Statistics type in IPv4validator LFB
   IPv6ValidatorStatsType    Statistics type in IPv6validator LFB
   IPv4UcastLPMStatsType     Statistics type in IPv4UcastLPM LFB
   IPv6UcastLPMStatsType     Statistics type in IPv6UcastLPM LFB
   QueueStatsType            Entry type for queue depth table
   MetadataDispatchType      Entry type for metadata dispatch table
```

[4.1.3](#). Compound Array

Compound array types are mostly created based on compound struct
types for LFB table components. The following compound array types
are defined in this base type library:

```
   Data Type Name               Brief Description
   --------------               -----------------
   EtherClassifyStatsTableType  Type for Ethernet classifier statistics
                                 information table.
   EtherDispatchTableType       Type for Ethernet dispatch table
   VlanInputTableType           Type for VLAN input table
   EncapTableType               Type for Ethernet encapsulation table
   IPv4PrefixTableType          Type for IPv4 prefix table
   IPv6PrefixTableType          Type for IPv6 prefix table
   IPv4NextHopTableType         Type for IPv4 next hop table
   IPv6NextHopTableType         Type for IPv6 next hop table
   MetadataDispatchTableType    Type for Metadata dispatch table
   QueueStatsTableType          Type for Queue depth table
```

4.2. Frame Types

   According to FE model [RFC5812], frame types are used in LFB
   definitions to define packet frame types both an LFB expects at its
   input port and the LFB emits at its output port. The <frameDef>
   element in the FE model is used to define a new frame type.

   The following frame types are defined in the base type library:

```
      Frame Name           Brief Description
      --------------       -----------------
      EthernetII           An Ethernet II frame
      ARP                  An ARP packet frame
      IPv4                 An IPv4 packet frame
      IPv6                 An IPv6 packet frame
      IPv4Unicast          An IPv4 unicast packet frame
      IPv4Multicast        An IPv4 multicast packet frame
      IPv6Unicast          An IPv6 unicast packet frame
      IPv6Multicast        An IPv6 multicast packet frame
      Arbitrary            Any type of packet frames
```

4.3. MetaData Types

   LFB Metadata is used to communicate per-packet state from one LFB to
   another. The <metadataDef> element in the FE model is used to define
   a new metadata type.

   The following metadata types are currently defined in the base type
   library.

```
   Metadata Name  Metadata ID  Brief Description
   ------------   ----------   -------------
   PHYPortID          1        Metadata indicating a physical port ID
   SrcMAC             2        Metadata indicating a source MAC address
   DstMAC             3        Metadata indicating a destination MAC
                                address.
   LogicalPortID      4        Metadata of a logical port ID
   EtherType          5        Metadata indicating an Ethernet type
   VlanID             6        Metadata of a VLAN ID
   VlanPriority       7        Metadata of a VLAN priority
   NextHopIPv4Addr    8        Metadata representing a next hop IPv4
                                address.
   NextHopIPv6Addr    9        Metadata representing a next hop IPv6
                                address.
   HopSelector        10       Metadata indicating a hop selector
   ExceptionID        11       Metadata indicating exception types for
                                exceptional cases during LFB processing.
   ValidateErrorID    12       Metadata indicating error types when a
                                packet passes validation process.
   L3PortID           13       Metadata indicating ID of an L3 logical
                                port.
   RedirectIndex      14       Metadata that CE sends to RedirectIn LFB,
                                indicating an associated packet a group
                                output port index of the LFB.
  MediaEncapInfoIndex 15       A search key a packet uses to look up a
                                table in related LFBs to select an
                                encapsulation media.
```

4.4. XML for Base Type Library

```
<?xml version="1.0" encoding="UTF-8"?>
<LFBLibrary xmlns="urn:ietf:params:xml:ns:forces:lfbmodel:1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    provides="BaseTypeLibrary">
  <frameDefs>
    <frameDef>
      <name>EthernetAll</name>
      <synopsis>Packet with any Ethernet type</synopsis>
    </frameDef>
    <frameDef>
      <name>EthernetII</name>
      <synopsis>Packet with Ethernet II type</synopsis>
    </frameDef>
    <frameDef>
      <name>ARP</name>
      <synopsis>ARP packet</synopsis>
    </frameDef>
```

```
       <frameDef>
          <name>IPv4</name>
          <synopsis>IPv4 packet</synopsis>
       </frameDef>
       <frameDef>
          <name>IPv6</name>
          <synopsis>IPv6 packet</synopsis>
       </frameDef>
       <frameDef>
          <name>IPv4Unicast</name>
          <synopsis>IPv4 unicast packet</synopsis>
       </frameDef>
       <frameDef>
          <name>IPv4Multicast</name>
          <synopsis>IPv4 multicast packet</synopsis>
       </frameDef>
       <frameDef>
          <name>IPv6Unicast</name>
          <synopsis>IPv6 unicast packet</synopsis>
       </frameDef>
       <frameDef>
          <name>IPv6Multicast</name>
          <synopsis>IPv6 multicast packet</synopsis>
       </frameDef>
       <frameDef>
          <name>Arbitrary</name>
          <synopsis>Any type of packet</synopsis>
       </frameDef>
    </frameDefs>
    <dataTypeDefs>
       <dataTypeDef>
          <name>IPv4Addr</name>
          <synopsis>IPv4 address</synopsis>
          <typeRef>byte[4]</typeRef>
       </dataTypeDef>
       <dataTypeDef>
          <name>IPv6Addr</name>
          <synopsis>IPv6 address</synopsis>
          <typeRef>byte[16]</typeRef>
       </dataTypeDef>
       <dataTypeDef>
          <name>IEEEMAC</name>
          <synopsis>IEEE MAC address</synopsis>
          <typeRef>byte[6]</typeRef>
       </dataTypeDef>
       <dataTypeDef>
         <name>LANSpeedType</name>
         <synopsis>LAN speed type</synopsis>
```

```
          <atomic>
           <baseType>uint32</baseType>
           <specialValues>
             <specialValue value="0x00000000">
              <name>LAN_SPEED_NONE</name>
              <synopsis>Nothing connected</synopsis>
             </specialValue>
             <specialValue value="0x00000001">
              <name>LAN_SPEED_10M</name>
              <synopsis>10M Ethernet</synopsis>
             </specialValue>
             <specialValue value="0x00000002">
              <name>LAN_SPEED_100M</name>
              <synopsis>100M Ethernet</synopsis>
             </specialValue>
             <specialValue value="0x00000003">
              <name>LAN_SPEED_1G</name>
              <synopsis>1G Ethernet</synopsis>
             </specialValue>
             <specialValue value="0x00000004">
              <name>LAN_SPEED_10G</name>
              <synopsis>10G Ethernet</synopsis>
             </specialValue>
             <specialValue value="0x00000005">
              <name>LAN_SPEED_40G</name>
              <synopsis>40G Ethernet</synopsis>
             </specialValue>
             <specialValue value="0x00000006">
              <name>LAN_SPEED_100G</name>
              <synopsis>100G Ethernet</synopsis>
             </specialValue>
             <specialValue value="0x00000007">
              <name>LAN_SPEED_400G</name>
              <synopsis>400G Ethernet</synopsis>
             </specialValue>
             <specialValue value="0x00000008">
              <name>LAN_SPEED_1T</name>
              <synopsis>1T Ethernet</synopsis>
             </specialValue>
             <specialValue value="0x00000009">
              <name>LAN_SPEED_OTHER</name>
              <synopsis>Other LAN speed type</synopsis>
             </specialValue>
             <specialValue value="0x0000000A">
              <name>LAN_SPEED_AUTO</name>
              <synopsis>LAN speed by auto negotiation</synopsis>
             </specialValue>
           </specialValues>
```

```
              </atomic>
          </dataTypeDef>
          <dataTypeDef>
            <name>DuplexType</name>
            <synopsis>Duplex mode type</synopsis>
            <atomic>
             <baseType>uint32</baseType>
             <specialValues>
               <specialValue value="0x00000001">
                <name>Auto</name>
                <synopsis>Auto negotiation</synopsis>
               </specialValue>
               <specialValue value="0x00000002">
                <name>HalfDuplex</name>
                <synopsis>Half duplex</synopsis>
               </specialValue>
               <specialValue value="0x00000003">
                <name>FullDuplex</name>
                <synopsis>Full duplex</synopsis>
               </specialValue>
             </specialValues>
            </atomic>
          </dataTypeDef>
          <dataTypeDef>
            <name>PortStatusType</name>
            <synopsis>
              Type for port status, used for both administrative and
              operative status.
            </synopsis>
            <atomic>
             <baseType>uchar</baseType>
             <specialValues>
               <specialValue value="0">
                <name>Disabled</name>
                <synopsis>Port disabled</synopsis>
               </specialValue>
               <specialValue value="1">
                <name>Up</name>
                <synopsis>Port up</synopsis>
               </specialValue>
               <specialValue value="2">
                <name>Down</name>
                <synopsis>Port down</synopsis>
               </specialValue>
             </specialValues>
            </atomic>
          </dataTypeDef>
          <dataTypeDef>
```

```
          <name>MACInStatsType</name>
          <synopsis>
            Data type defined for statistics in EtherMACIn LFB.
          </synopsis>
          <struct>
             <component componentID="1">
                <name>NumPacketsReceived</name>
                <synopsis>Number of packets received</synopsis>
                <typeRef>uint64</typeRef>
             </component>
             <component componentID="2">
                <name>NumPacketsDropped</name>
                <synopsis>Number of packets dropped</synopsis>
                <typeRef>uint64</typeRef>
             </component>
          </struct>
       </dataTypeDef>
       <dataTypeDef>
          <name>MACOutStatsType</name>
          <synopsis>
            Data type defined for statistics in EtherMACOut LFB.
          </synopsis>
          <struct>
             <component componentID="1">
                <name>NumPacketsTransmitted</name>
                <synopsis>Number of packets transmitted</synopsis>
                <typeRef>uint64</typeRef>
             </component>
             <component componentID="2">
                <name>NumPacketsDropped</name>
                <synopsis>Number of packets dropped</synopsis>
                <typeRef>uint64</typeRef>
             </component>
          </struct>
       </dataTypeDef>
       <dataTypeDef>
          <name>EtherDispatchEntryType</name>
          <synopsis>
            Data type defined for entry of Ethernet dispatch
            table in EtherClassifier LFB.
          </synopsis>
          <struct>
             <component componentID="1">
                <name>LogicalPortID</name>
                <synopsis>Logical port ID</synopsis>
                <typeRef>uint32</typeRef>
             </component>
             <component componentID="2">
```

```
              <name>EtherType</name>
              <synopsis>
               The Ethernet type of the Ethernet packet.
              </synopsis>
              <typeRef>uint16</typeRef>
           </component>
           <component componentID="3">
              <name>Reserved</name>
              <synopsis>
              A reserved bit space mainly for purpose of padding
              and packing efficiency.
              </synopsis>
              <typeRef>uint16</typeRef>
           </component>
           <component componentID="4">
              <name>LFBOutputSelectIndex</name>
               <synopsis>
                 Index for a packet to select an instance in the
                 group output port of EtherClassifier LFB to output.
               </synopsis>
              <typeRef>uint32</typeRef>
           </component>
        </struct>
     </dataTypeDef>
     <dataTypeDef>
        <name>EtherDispatchTableType</name>
        <synopsis>
          Data type defined for Ethernet dispatch table in
          EtherClassifier LFB. The table is composed of an array
          of entries with EtherDispatchEntryType data type.
        </synopsis>
        <array type="variable-size">
          <typeRef>EtherDispatchEntryType</typeRef>
        </array>
     </dataTypeDef>
     <dataTypeDef>
        <name>VlanIDType</name>
        <synopsis>Data type for VLAN ID</synopsis>
        <atomic>
        <baseType>uint16</baseType>
          <rangeRestriction>
             <allowedRange min="0" max="4095"/>
          </rangeRestriction>
        </atomic>
      </dataTypeDef>
     <dataTypeDef>
        <name>VlanPriorityType</name>
        <synopsis>Data type for VLAN priority</synopsis>
```

```
        <atomic>
        <baseType>uchar</baseType>
          <rangeRestriction>
             <allowedRange min="0" max="7"/>
          </rangeRestriction>
        </atomic>
    </dataTypeDef>
    <dataTypeDef>
        <name>VlanInputTableEntryType</name>
        <synopsis>
          Data type for entry of VLAN input table in EtherClassifier
          LFB. Each entry of the table contains an incoming port ID,
          a VLAN ID and a logical port ID. Every input packet is
          assigned with a new logical port ID according to the
          packet incoming port ID and the VLAN ID.
          </synopsis>
        <struct>
          <component componentID="1">
             <name>IncomingPortID</name>
             <synopsis>The incoming port ID</synopsis>
             <typeRef>uint32</typeRef>
          </component>
          <component componentID="2">
             <name>VlanID</name>
             <synopsis>The VLAN ID</synopsis>
             <typeRef>VlanIDType</typeRef>
          </component>
          <component componentID="3">
             <name>Reserved</name>
             <synopsis>
             A reserved bit space mainly for purpose of padding
             and packing efficiency.
             </synopsis>
             <typeRef>uint16</typeRef>
          </component>
          <component componentID="4">
             <name>LogicalPortID</name>
             <synopsis>The logical port ID</synopsis>
             <typeRef>uint32</typeRef>
          </component>
        </struct>
    </dataTypeDef>
    <dataTypeDef>
        <name>VlanInputTableType</name>
        <synopsis>
          Data type for the VLAN input table in EtherClassifier
          LFB. The table is composed of an array of entries with
          VlanInputTableEntryType.
```

```
        </synopsis>
        <array type="variable-size">
          <typeRef>VlanInputTableEntryType</typeRef>
        </array>
      </dataTypeDef>
      <dataTypeDef>
        <name>EtherClassifyStatsType</name>
        <synopsis>
          Data type for entry of statistics table in EtherClassifier
          LFB.
        </synopsis>
        <struct>
          <component componentID="1">
            <name>EtherType</name>
            <synopsis>
             The Ethernet type of the Ethernet packet.
            </synopsis>
            <typeRef>uint16</typeRef>
          </component>
          <component componentID="2">
            <name>Reserved</name>
            <synopsis>
            A reserved bit space mainly for purpose of padding
            and packing efficiency.
            </synopsis>
            <typeRef>uint16</typeRef>
          </component>
          <component componentID="3">
            <name>PacketsNum</name>
            <synopsis>Packets number</synopsis>
            <typeRef>uint64</typeRef>
          </component>
        </struct>
      </dataTypeDef>
      <dataTypeDef>
        <name>EtherClassifyStatsTableType</name>
        <synopsis>
          Data type for statistics table in EtherClassifier LFB.
        </synopsis>
        <array type="variable-size">
          <typeRef>EtherClassifyStatsType</typeRef>
        </array>
      </dataTypeDef>
      <dataTypeDef>
        <name>IPv4ValidatorStatsType</name>
        <synopsis>
          Data type for statistics in IPv4validator LFB.
        </synopsis>
```

```
            <struct>
               <component componentID="1">
                  <name>badHeaderPkts</name>
                  <synopsis>Number of packets with bad header</synopsis>
                  <typeRef>uint64</typeRef>
               </component>
               <component componentID="2">
                  <name>badTotalLengthPkts</name>
                  <synopsis>
                     Number of packets with bad total length
                  </synopsis>
                  <typeRef>uint64</typeRef>
               </component>
               <component componentID="3">
                  <name>badTTLPkts</name>
                  <synopsis>Number of packets with bad TTL</synopsis>
                  <typeRef>uint64</typeRef>
               </component>
               <component componentID="4">
                  <name>badChecksumPkts</name>
                  <synopsis>Number of packets with bad checksum</synopsis>
                  <typeRef>uint64</typeRef>
               </component>
            </struct>
         </dataTypeDef>
         <dataTypeDef>
            <name>IPv6ValidatorStatsType</name>
            <synopsis>
              Data type for statistics in IPv6validator LFB.
            </synopsis>
            <struct>
               <component componentID="1">
                  <name>badHeaderPkts</name>
                  <synopsis>Number of packets with bad header</synopsis>
                  <typeRef>uint64</typeRef>
               </component>
               <component componentID="2">
                  <name>badTotalLengthPkts</name>
                  <synopsis>
                  Number of packets with bad total length.
                  </synopsis>
                  <typeRef>uint64</typeRef>
               </component>
               <component componentID="3">
                  <name>badHopLimitPkts</name>
                  <synopsis>
                  Number of packets with bad hop limit.
                  </synopsis>
```

```
                    <typeRef>uint64</typeRef>
                 </component>
              </struct>
           </dataTypeDef>
           <dataTypeDef>
              <name>IPv4PrefixInfoType</name>
              <synopsis>Data type for entry of IPv4 longest prefix match
               table in IPv4UcastLPM LFB. 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>
              <struct>
                 <component componentID="1">
                    <name>IPv4Address</name>
                    <synopsis>The destination IPv4 address</synopsis>
                    <typeRef>IPv4Addr</typeRef>
                 </component>
                 <component componentID="2">
                    <name>Prefixlen</name>
                    <synopsis>The prefix length</synopsis>
                    <atomic>
                       <baseType>uchar</baseType>
                       <rangeRestriction>
                          <allowedRange min="0" max="32"/>
                       </rangeRestriction>
                    </atomic>
                 </component>
                 <component componentID="3">
                    <name>ECMPFlag</name>
                    <synopsis>The ECMP flag</synopsis>
                    <atomic>
                       <baseType>boolean</baseType>
                       <specialValues>
                          <specialValue value="false">
                             <name>False</name>
                             <synopsis>
                              ECMP false, indicating the route
                              does not have multiple next hops.
                             </synopsis>
                          </specialValue>
                          <specialValue value="true">
                             <name>True</name>
                             <synopsis>
                               ECMP true, indicating the route
                               has multiple next hops.
                             </synopsis>
                          </specialValue>
                       </specialValues>
                    </atomic>
```

```
              </component>
              <component componentID="4">
                 <name>DefaultRouteFlag</name>
                 <synopsis>Default route flag</synopsis>
                 <atomic>
                    <baseType>boolean</baseType>
                    <specialValues>
                       <specialValue value="false">
                          <name>False</name>
                          <synopsis>
                            Default route false, indicating the
                            route is not a default route.
                          </synopsis>
                       </specialValue>
                       <specialValue value="true">
                          <name>True</name>
                          <synopsis>
                            Default route true, indicating the
                            route is a default route.
                          </synopsis>
                       </specialValue>
                    </specialValues>
                 </atomic>
              </component>
              <component componentID="5">
                 <name>Reserved</name>
                 <synopsis>
                 A reserved bit space mainly for purpose of padding
                 and packing efficiency.
                 </synopsis>
                 <typeRef>uchar</typeRef>
              </component>
              <component componentID="6">
                 <name>HopSelector</name>
                 <synopsis>
                   The HopSelector produced by the prefix matching LFB,
                   which will be output to downstream LFB to find next
                   hop information.
                 </synopsis>
                 <typeRef>uint32</typeRef>
              </component>
           </struct>
        </dataTypeDef>
        <dataTypeDef>
           <name>IPv4PrefixTableType</name>
           <synopsis>
             Data type for IPv4 longest prefix match table in
             IPv4UcastLPM LFB. Entry of the table is
```

```
          of IPv4PrefixInfoType data type.
      </synopsis>
      <array type="variable-size">
        <typeRef>IPv4PrefixInfoType</typeRef>
      </array>
   </dataTypeDef>
   <dataTypeDef>
      <name>IPv4UcastLPMStatsType</name>
      <synopsis>
       Data type for statistics in IPv4UcastLPM LFB.
      </synopsis>
      <struct>
         <component componentID="1">
            <name>InRcvdPkts</name>
            <synopsis>Number of received input packets.</synopsis>
            <typeRef>uint64</typeRef>
         </component>
         <component componentID="2">
            <name>FwdPkts</name>
            <synopsis>Number of forwarded packets.</synopsis>
            <typeRef>uint64</typeRef>
         </component>
         <component componentID="3">
            <name>NoRoutePkts</name>
            <synopsis>
             Number of packets with no route found.
            </synopsis>
            <typeRef>uint64</typeRef>
         </component>
      </struct>
   </dataTypeDef>
   <dataTypeDef>
      <name>IPv6PrefixInfoType</name>
      <synopsis>Data type for entry of IPv6 longest prefix match
       table in IPv6UcastLPM LFB. The destination IPv6 address
       of every input packet is used as a search key to look up
       the table to find out a next hop selector.</synopsis>
      <struct>
         <component componentID="1">
            <name>IPv6Address</name>
            <synopsis>The destination IPv6 address</synopsis>
            <typeRef>IPv6Addr</typeRef>
         </component>
         <component componentID="2">
            <name>Prefixlen</name>
            <synopsis>The prefix length</synopsis>
            <atomic>
                <baseType>uchar</baseType>
```

```
            <rangeRestriction>
               <allowedRange min="0" max="32"/>
            </rangeRestriction>
         </atomic>
      </component>
      <component componentID="3">
         <name>ECMPFlag</name>
         <synopsis>ECMP flag</synopsis>
         <atomic>
            <baseType>boolean</baseType>
            <specialValues>
               <specialValue value="false">
                  <name>False</name>
                  <synopsis>ECMP false</synopsis>
               </specialValue>
               <specialValue value="true">
                  <name>True</name>
                  <synopsis>ECMP true</synopsis>
               </specialValue>
            </specialValues>
         </atomic>
      </component>
      <component componentID="4">
         <name>DefaultRouteFlag</name>
         <synopsis>Default route flag</synopsis>
         <atomic>
            <baseType>boolean</baseType>
            <specialValues>
               <specialValue value="false">
                  <name>False</name>
                  <synopsis>Default false</synopsis>
               </specialValue>
               <specialValue value="true">
                  <name>True</name>
                  <synopsis>Default route true</synopsis>
               </specialValue>
            </specialValues>
         </atomic>
      </component>
      <component componentID="5">
         <name>Reserved</name>
         <synopsis>
         A reserved bit space mainly for purpose of padding
         and packing efficiency.
         </synopsis>
         <typeRef>uchar</typeRef>
      </component>
      <component componentID="6">
```

```
               <name>HopSelector</name>
               <synopsis>
                 The HopSelector produced by the prefix matching LFB,
                 which will be output to downstream LFB to find next
                 hop information.
               </synopsis>
               <typeRef>uint32</typeRef>
            </component>
         </struct>
      </dataTypeDef>
      <dataTypeDef>
         <name>IPv6PrefixTableType</name>
         <synopsis>
           Data type for IPv6 longest prefix match table in
           IPv6UcastLPM LFB. Entry of the table is
           of IPv6PrefixInfoType data type.
         </synopsis>
         <array type="variable-size">
           <typeRef>IPv6PrefixInfoType</typeRef>
         </array>
      </dataTypeDef>
      <dataTypeDef>
         <name>IPv6UcastLPMStatsType</name>
         <synopsis>Data type for statistics in IPv6UcastLPM LFB
         </synopsis>
         <struct>
            <component componentID="1">
               <name>InRcvdPkts</name>
               <synopsis>Number of received input packets</synopsis>
               <typeRef>uint64</typeRef>
            </component>
            <component componentID="2">
               <name>FwdPkts</name>
               <synopsis>Number of forwarded packets</synopsis>
               <typeRef>uint64</typeRef>
            </component>
            <component componentID="3">
               <name>NoRoutePkts</name>
               <synopsis>
                Number of packets with no route found.
               </synopsis>
               <typeRef>uint64</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>L3PortID</name>
                <synopsis>
                 The ID of the logical output port that is to pass
                 onto downstream LFB, indicating what port to the
                 neighbor is as defined by L3.
                </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>IPv4Addr</typeRef>
            </component>
            <component componentID="4">
                <name>MediaEncapInfoIndex</name>
                <synopsis>
                  The index passed onto a downstream encapsulation
                  LFB, used there as a search key to lookup further
                  encapsulation information.
                </synopsis>
                <typeRef>uint32</typeRef>
            </component>
            <component componentID="5">
                <name>LFBOutputSelectIndex</name>
                 <synopsis>
                   The index for the IPv4NextHop LFB to choose an
                   instance in the group output port of the LFB to
                   output.
                 </synopsis>
                <typeRef>uint32</typeRef>
            </component>
          </struct>
      </dataTypeDef>
      <dataTypeDef>
```

```
            <name>IPv4NextHopTableType</name>
            <synopsis>
              Data type for IPv4 next hop table in IPv4NextHop LFB.
              Entry of the table is of IPv4NextHopInfoType data type.
            </synopsis>
            <array type="variable-size">
              <typeRef>IPv4NextHopInfoType</typeRef>
            </array>
        </dataTypeDef>
        <dataTypeDef>
            <name>IPv6NextHopInfoType</name>
            <synopsis>
              Data type for entry of IPv6 next hop information table
              in IPv6NextHop 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>L3PortID</name>
                  <synopsis>
                   The ID of the logical output port that is to pass
                   onto downstream LFB, indicating what port to the
                   neighbor is as defined by L3.
                  </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 IPv6 address</synopsis>
                  <typeRef>IPv6Addr</typeRef>
              </component>
              <component componentID="4">
                  <name>MediaEncapInfoIndex</name>
                  <synopsis>
                    The index passed onto a downstream encapsulation
                    LFB, used there as a search key to lookup further
                    encapsulation information.
                  </synopsis>
                  <typeRef>uint32</typeRef>
              </component>
```

```
            <component componentID="5">
               <name>LFBOutputSelectIndex</name>
                <synopsis>
                 The index for the IPv6NextHop LFB to choose an instance
                 in the group output port of the LFB to output.
                </synopsis>
               <typeRef>uint32</typeRef>
            </component>
         </struct>
      </dataTypeDef>
      <dataTypeDef>
         <name>IPv6NextHopTableType</name>
         <synopsis>
           Data type for IPv6 next hop table in IPv6NextHop LFB.
           Entry of the table is of IPv6NextHopInfoType data type.
         </synopsis>
         <array type="variable-size">
           <typeRef>IPv6NextHopInfoType</typeRef>
         </array>
      </dataTypeDef>
      <dataTypeDef>
         <name>EncapTableEntryType</name>
         <synopsis>
           Data type for entry of Ethernet encapsulation table in
           EtherEncap LFB. The LFB uses the MediaEncapInfoIndex
           received from upstream LFB as index of the table to
           find encapsulation information of every packet.
         </synopsis>
         <struct>
            <component componentID="1">
               <name>DstMac</name>
               <synopsis>
                 Destination MAC address for Ethernet encapsulation of
                 the packet.
               </synopsis>
               <typeRef>IEEEMAC</typeRef>
            </component>
            <component componentID="2">
               <name>SrcMac</name>
               <synopsis>
                 Source MAC address for Ethernet encapsulation of the
                 packet.
               </synopsis>
               <typeRef>IEEEMAC</typeRef>
            </component>
            <component componentID="3">
               <name>VlanID</name>
               <synopsis>The VLAN ID assigned to the packet</synopsis>
```

```
                <typeRef>VlanIDType</typeRef>
            </component>
             <component componentID="4">
               <name>Reserved</name>
               <synopsis>
                A reserved bit space mainly for purpose of padding
                and packing efficiency.
               </synopsis>
               <typeRef>uint16</typeRef>
            </component>
            <component componentID="5">
               <name>L2PortID</name>
               <synopsis>
                  The L2 logical output port ID for the packet.
               </synopsis>
               <typeRef>uint32</typeRef>
            </component>
         </struct>
      </dataTypeDef>
      <dataTypeDef>
         <name>EncapTableType</name>
         <synopsis>
           Data type for Ethernet encapsulation table in Etherencap
           LFB. Entry of the table is of EncapTableEntryType data
           type.
         </synopsis>
         <array type="variable-size">
           <typeRef>EncapTableEntryType</typeRef>
         </array>
      </dataTypeDef>
      <dataTypeDef>
         <name>MetadataDispatchType</name>
         <synopsis>
           Data type for entry of metadata dispatch table used in
           BasicMetadataDispatch LFB. The LFB uses a metadata value
           as a search key to look up the table to find an index of
           the LFB group output port to output the packet.
         </synopsis>
         <struct>
            <component componentID="1">
               <name>MetadataValue</name>
               <synopsis>The value of the dispatch metadata</synopsis>
               <typeRef>uint32</typeRef>
            </component>
            <component componentID="2">
               <name>OutputIndex</name>
               <synopsis>
                  Index of a group output port for outgoing packets.
```

```
              </synopsis>
              <typeRef>uint32</typeRef>
            </component>
         </struct>
      </dataTypeDef>
      <dataTypeDef>
         <name>MetadataDispatchTableType</name>
         <synopsis>
           Data type for metadata dispatch table used in
           BasicMetadataDispatch LFB. Metadata value of
           the table is also defined as a content key field.
         </synopsis>
         <array type="variable-size">
           <typeRef>MetadataDispatchType</typeRef>
           <contentKey contentKeyID="1">
           <contentKeyField>MetadataValue</contentKeyField>
           </contentKey>
         </array>
      </dataTypeDef>
      <dataTypeDef>
         <name>SchdDisciplineType</name>
         <synopsis>Scheduling discipline type</synopsis>
         <atomic>
            <baseType>uint32</baseType>
            <specialValues>
               <specialValue value="1">
                  <name>RR</name>
                  <synopsis>
                    Round Robin scheduling discipline
                  </synopsis>
               </specialValue>
            </specialValues>
         </atomic>
      </dataTypeDef>
      <dataTypeDef>
         <name>QueueStatsType</name>
         <synopsis>
           Data type for entry of queue statistics table in
           GenericScheduler LFB.
         </synopsis>
         <struct>
           <component componentID="1">
              <name>QueueID</name>
              <synopsis>The input queue ID</synopsis>
              <typeRef>uint32</typeRef>
           </component>
           <component componentID="2">
              <name>QueueDepthInPackets</name>
```

```
              <synopsis>Current queue depth in packets</synopsis>
              <typeRef>uint32</typeRef>
          </component>
          <component componentID="3">
              <name>QueueDepthInBytes</name>
              <synopsis>Current queue depth in bytes</synopsis>
              <typeRef>uint32</typeRef>
          </component>
        </struct>
    </dataTypeDef>
    <dataTypeDef>
        <name>QueueStatsTableType</name>
        <synopsis>
          Data type for queue statistics table in GenericScheduler
          LFB. Entry of the table is of QueueStatsType data type.
        </synopsis>
        <array type="variable-size">
          <typeRef>QueueStatsType</typeRef>
        </array>
    </dataTypeDef>
  </dataTypeDefs>
  <metadataDefs>
    <metadataDef>
        <name>PHYPortID</name>
        <synopsis>Metadata indicating physical port ID</synopsis>
        <metadataID>1</metadataID>
        <typeRef>uint32</typeRef>
    </metadataDef>
    <metadataDef>
        <name>SrcMAC</name>
        <synopsis>Metadata indicating source MAC address</synopsis>
        <metadataID>2</metadataID>
        <typeRef>IEEEMAC</typeRef>
    </metadataDef>
    <metadataDef>
        <name>DstMAC</name>
        <synopsis>
          Metadata indicating destination MAC address.
        </synopsis>
        <metadataID>3</metadataID>
        <typeRef>IEEEMAC</typeRef>
    </metadataDef>
    <metadataDef>
        <name>LogicalPortID</name>
        <synopsis>Metadata of logical port ID</synopsis>
        <metadataID>4</metadataID>
        <typeRef>uint32</typeRef>
    </metadataDef>
```

```
        <metadataDef>
           <name>EtherType</name>
           <synopsis>Metadata indicating Ethernet type</synopsis>
           <metadataID>5</metadataID>
           <typeRef>uint16</typeRef>
        </metadataDef>
        <metadataDef>
           <name>VlanID</name>
           <synopsis>Metadata of VLAN ID</synopsis>
           <metadataID>6</metadataID>
           <typeRef>VlanIDType</typeRef>
        </metadataDef>
        <metadataDef>
           <name>VlanPriority</name>
           <synopsis>Metadata of VLAN priority</synopsis>
           <metadataID>7</metadataID>
           <typeRef>VlanPriorityType</typeRef>
        </metadataDef>
        <metadataDef>
           <name>NextHopIPv4Addr</name>
           <synopsis>
             Metadata representing a next hop IPv4 address
           </synopsis>
           <metadataID>8</metadataID>
           <typeRef>IPv4Addr</typeRef>
        </metadataDef>
        <metadataDef>
           <name>NextHopIPv6Addr</name>
           <synopsis>
             Metadata representing a next hop IPv6 address
           </synopsis>
           <metadataID>9</metadataID>
           <typeRef>IPv6Addr</typeRef>
        </metadataDef>
        <metadataDef>
           <name>HopSelector</name>
           <synopsis>Metadata indicating a hop selector</synopsis>
           <metadataID>10</metadataID>
           <typeRef>uint32</typeRef>
        </metadataDef>
        <metadataDef>
           <name>ExceptionID</name>
           <synopsis>
             Metadata indicating exception types for exceptional cases
             during packet processing.
           </synopsis>
           <metadataID>11</metadataID>
           <atomic>
```

```
                <baseType>uint32</baseType>
                <specialValues>
                    <specialValue value="0">
                      <name>AnyUnrecognizedExceptionCase</name>
                      <synopsis>Any unrecognized exception case</synopsis>
                      </specialValue>
                    <specialValue value="1">
                      <name>ClassifyNoMatching</name>
                      <synopsis>
                       Exception case: no matching of tables in
                       EtherClassifier LFB.
                      </synopsis>
                    </specialValue>
                    <specialValue value="2">
                      <name>MediaEncapInfoIndexInvalid</name>
                      <synopsis>
                       Exception case: the MediaEncapInfoIndex value of
                       the packet is invalid and cannot be allocated in
                       the EncapTable in EtherEncap LFB.
                      </synopsis>
                    </specialValue>
                    <specialValue value="3">
                      <name>EncapTableLookupFailed</name>
                      <synopsis>
                       Exception case: the packet fails lookup of the
                       EncapTable table in EtherEncap LFB even though the
                       MediaEncapInfoIndex is valid.
                      </synopsis>
                    </specialValue>
                    <specialValue value="4">
                      <name>BadTTL</name>
                      <synopsis>
                       Exception case: packet with expired TTL
                      </synopsis>
                    </specialValue>
                    <specialValue value="5">
                      <name>IPv4HeaderLengthMismatch</name>
                      <synopsis>
                       Exception case: packet with header length more
                       than 5 words.
                      </synopsis>
                    </specialValue>
                    <specialValue value="6">
                        <name>RouterAlertOptions</name>
                        <synopsis>
                         Exception case: packet IP head includes router
                         alert options.
                        </synopsis>
```

```
                    </specialValue>
                    <specialValue value="7">
                       <name>IPv6HopLimitZero</name>
                       <synopsis>
                        Exception case: packet with the hop limit to zero.
                       </synopsis>
                    </specialValue>
                    <specialValue value="8">
                       <name>IPv6NextHeaderHBH</name>
                       <synopsis>
                        Exception case: packet with next header set to
                        Hop-by-Hop.
                       </synopsis>
                    </specialValue>
                    <specialValue value="9">
                       <name>SrcAddressExecption</name>
                       <synopsis>
                        Exception case: packet with exceptional source
                        address.
                       </synopsis>
                    </specialValue>
                    <specialValue value="10">
                       <name>DstAddressExecption</name>
                       <synopsis>
                        Exception case: packet with exceptional destination
                        address.
                       </synopsis>
                    </specialValue>
                    <specialValue value="11">
                       <name>LPMLookupFailed</name>
                       <synopsis>
                        Exception case: packet failed the LPM table lookup
                        in a prefix match LFB.
                       </synopsis>
                    </specialValue>
                    <specialValue value="12">
                       <name>HopSelectorInvalid</name>
                       <synopsis>
                        Exception case: HopSelector for the packet is
                        invalid.
                       </synopsis>
                    </specialValue>
                    <specialValue value="13">
                       <name>NextHopLookupFailed</name>
                       <synopsis>
                        Exception case: packet failed lookup of a next hop
                        table even though HopSelector is valid.
                       </synopsis>
```

```
                    </specialValue>
                    <specialValue value="14">
                       <name>FragRequired</name>
                       <synopsis>
                        Exception case: packet fragmentation is required
                       </synopsis>
                    </specialValue>
                    <specialValue value="15">
                       <name>MetadataNoMatching</name>
                       <synopsis>
                        Exception case: there is no matching when looking
                        up the metadata dispatch table in
                        BasicMetadataDispatch LFB.
                       </synopsis>
                    </specialValue>
                 </specialValues>
              </atomic>
        </metadataDef>
        <metadataDef>
            <name>ValidateErrorID</name>
            <synopsis>
              Metadata indicating error types when a packet passes
              validation process.
            </synopsis>
            <metadataID>12</metadataID>
            <atomic>
               <baseType>uint32</baseType>
               <specialValues>
                  <specialValue value="0">
                     <name>AnyUnrecognizedValidateErrorCase</name>
                     <synopsis>
                       Any unrecognized validate error case.
                     </synopsis>
                  </specialValue>
                  <specialValue value="1">
                     <name>InvalidIPv4PacketSize</name>
                     <synopsis>
                      Error case: packet length reported by the link
                      layer is less than 20 bytes.
                     </synopsis>
                  </specialValue>
                  <specialValue value="2">
                     <name>NotIPv4Packet</name>
                     <synopsis>
                      Error case: packet is not IP version 4</synopsis>
                  </specialValue>
                  <specialValue value="3">
                     <name>InvalidIPv4HeaderLengthSize</name>
```

```
                    <synopsis>
                     Error case: packet with header length field in
                     the header less than 5 words.
                    </synopsis>
                 </specialValue>
                 <specialValue value="4">
                    <name>InvalidIPv4LengthFieldSize</name>
                    <synopsis>
                     Error case: packet with total length field in the
                     header less than 20 bytes.
                    </synopsis>
                 </specialValue>
                 <specialValue value="5">
                    <name>InvalidIPv4Checksum</name>
                    <synopsis>
                     Error case: packet with invalid checksum.
                     </synopsis>
                 </specialValue>
                 <specialValue value="6">
                    <name>InvalidIPv4SrcAddr</name>
                    <synopsis>
                     Error case: packet with invalid IPv4 source
                     address.
                    </synopsis>
                 </specialValue>
                 <specialValue value="7">
                    <name>InvalidIPv4DstAddr</name>
                    <synopsis>
                     Error case: packet with invalid IPv4 destination
                     address.
                    </synopsis>
                 </specialValue>
                 <specialValue value="8">
                    <name>InvalidIPv6PacketSize</name>
                    <synopsis>
                     Error case: packet size is less than 40 bytes.
                    </synopsis>
                 </specialValue>
                 <specialValue value="9">
                    <name>NotIPv6Packet</name>
                    <synopsis>
                     Error case: packet is not IP version 6
                     </synopsis>
                 </specialValue>
                 <specialValue value="10">
                    <name>InvalidIPv6SrcAddr</name>
                    <synopsis>
                     Error case: packet with invalid IPv6 source address.
```

```
                    </synopsis>
                </specialValue>
                <specialValue value="11">
                    <name>InvalidIPv6DstAddr</name>
                    <synopsis>
                     Error case: packet with invalid IPv6 destination
                     address.
                    </synopsis>
                </specialValue>
            </specialValues>
          </atomic>
      </metadataDef>
      <metadataDef>
          <name>L3PortID</name>
          <synopsis>
            Metadata indicating ID of an L3 logical port
          </synopsis>
          <metadataID>13</metadataID>
          <typeRef>uint32</typeRef>
      </metadataDef>
      <metadataDef>
          <name>RedirectIndex</name>
          <synopsis>
            Metadata that CE sends to RedirectIn LFB, indicating
            the index of the LFB group output port.
          </synopsis>
          <metadataID>14</metadataID>
          <typeRef>uint32</typeRef>
      </metadataDef>
      <metadataDef>
          <name>MediaEncapInfoIndex</name>
          <synopsis>
            A search key a packet uses to look up a table to select
            an encapsulation media.
          </synopsis>
          <metadataID>15</metadataID>
          <typeRef>uint32</typeRef>
      </metadataDef>
    </metadataDefs>
</LFBLibrary>
```

5. LFB Class Description

   According to ForCES specifications, LFB (Logical Function Block) is a
   well defined, logically separable functional block that resides in an
   FE, and is a functionally accurate abstraction of the FE's processing
   capabilities. An LFB Class (or type) is a template that represents a
   fine-grained, logically separable aspect of FE processing. Most LFBs
   are related to packet processing in the data path. LFB classes are
   the basic building blocks of the FE model. Note that [RFC5810] has
   already defined an 'FE Protocol LFB' which is a logical entity in
   each FE to control the ForCES protocol. [RFC5812] has already
   defined an 'FE Object LFB'. Information like the FE Name, FE ID, FE
   State, LFB Topology in the FE are represented in this LFB.

   As specified in Section 3.1, this document focuses on the base LFB
   library for implementing typical router functions, especially for IP
   forwarding functions. As a result, LFB classes in the library are
   all base LFBs to implement router forwarding.

   In this section, the terms "upstream LFB" and "downstream LFB" are
   used. These are used relative to an LFB to an LFB that is being
   described. An "upstream LFB" is one whose output ports are connected
   to input ports of the LFB under consideration such that output
   (typically packets with metadata) can be sent from the "upstream LFB"
   to the LFB under consideration. Similarly, a "downstream LFB" whose
   input ports are connected to output ports of the LFB under
   consideration such that the LFB under consideration can send
   information to the "downstream LFB". Note that in some rare
   topologies, an LFB may be both upstream and downstream relative to
   another LFB.

   Also note that, as a default provision of [RFC5812], in FE model, all
   metadata produced by upstream LFBs will pass through all downstream
   LFBs by default without being specified by input port or output port.
   Only those metadata that will be used (consumed) by an LFB will be
   explicitly marked in input of the LFB as expected metadata. For
   instance, in downstream LFBs of a physical layer LFB, even there is
   no specific metadata expected, metadata like PHYPortID produced by
   the physical layer LFB will always pass through all downstream LFBs
   regardless of whether the metadata has been expected by the LFBs or
   not.

5.1. Ethernet Processing LFBs

   As the most popular physical and data link layer protocols, Ethernet
   is widely deployed. It becomes a basic requirement for a router to
   be able to process various Ethernet data packets.

Note that there exist different versions of Ethernet formats, like
Ethernet V2, 802.3 RAW, IEEE 802.3/802.2, IEEE 802.3/802.2 SNAP.
There also exist varieties of LAN techniques based on Ethernet, like
various VLANs, MACinMAC, etc. Ethernet processing LFBs defined here
are intended to be able to cope with all these variations of Ethernet
technology.

There are also various types of Ethernet physical interface media.
Among them, copper and fiber media may be the most popular ones. As
a base LFB definition and a starting point, the document only defines
an Ethernet physical LFB with copper media. For other media
interfaces, specific LFBs may be defined in the future versions of
the library.

5.1.1. EtherPHYCop

EtherPHYCop LFB abstracts an Ethernet interface physical layer with
media limited to copper.

5.1.1.1. Data Handling

This LFB is the interface to the Ethernet physical media. The LFB
handles Ethernet frames coming in from or going out of the FE.
Ethernet frames sent and received cover all packets encapsulated with
different versions of Ethernet protocols, like Ethernet V2, 802.3
RAW, IEEE 802.3/802.2,IEEE 802.3/802.2 SNAP, including packets
encapsulated with varieties of LAN techniques based on Ethernet, like
various VLANs, MACinMAC, etc. Therefore in the XML an EthernetAll
frame type has been introduced.

Ethernet frames are received from the physical media port and passed
downstream to LFBs such as EtherMACIn via a singleton output known as
"EtherPHYOut". A 'PHYPortID' metadata, to indicate which physical
port the frame came into from the external world, is passed along
with the frame.

Ethernet packets are received by this LFB from upstream LFBs such as
EtherMacOut LFBs via the singleton input known as "EtherPHYIn" before
being sent out onto the external world.

5.1.1.2. Components

The AdminStatus component is defined for CE to administratively
manage the status of the LFB. The CE may administratively startup or
shutdown the LFB by changing the value of AdminStatus. The default
value is set to 'Down'.

An OperStatus component captures the physical port operational

status. A PHYPortStatusChanged event is defined so the LFB can
report to the CE whenever there is an operational status change of
the physical port.

The PHYPortID component is a unique identification for a physical
port. It is defined as 'read-only' by CE. Its value is enumerated
by FE. The component will be used to produce a 'PHYPortID' metadata
at the LFB output and to associate it to every Ethernet packet this
LFB receives. The metadata will be handed to downstream LFBs for
them to use the PHYPortID.

A group of components are defined for link speed management. The
AdminLinkSpeed is for CE to configure link speed for the port and the
OperLinkSpeed is for CE to query the actual link speed in operation.
The default value for the AdminLinkSpeed is set to auto-negotiation
mode.

A group of components are defined for duplex mode management. The
AdminDuplexMode is for CE to configure proper duplex mode for the
port and the OperDuplexMode is for CE to query the actual duplex mode
in operation. The default value for the AdminDuplexMode is set to
auto-negotiation mode.

A CarrierStatus component captures the status of the carrier and
specifies whether the port link is operationally up. The default
value for the CarrierStatus is 'false'.

5.1.1.3. Capabilities

The capability information for this LFB includes the link speeds that
are supported by the FE (SupportedLinkSpeed) as well as the supported
duplex modes (SupportedDuplexMode).

5.1.1.4. Events

Several events are generated. There is an event for changes in the
status of the physical port (PhyPortStatusChanged). Such an event
will notify that the physical port status has been changed and the
report will include the new status of the physical port.

Another event captures changes in the operational link speed
(LinkSpeedChanged). Such an event will notify the CE that the
operational speed has been changed and the report will include the
new negotiated operational speed.

A final event captures changes in the duplex mode
(DuplexModeChanged). Such an event will notify the CE that the
duplex mode has been changed and the report will include the new

negotiated duplex mode.

5.1.2. EtherMACIn

EtherMACIn LFB abstracts an Ethernet port at MAC data link layer.
This LFB describes Ethernet processing functions like MAC address
locality check, deciding if the Ethernet packets should be bridged,
providing Ethernet layer flow control, etc.

5.1.2.1. Data Handling

The LFB is expected to receive all types of Ethernet packets, via a
singleton input known as "EtherPktsIn", which are usually output from
some Ethernet physical layer LFB, like an EtherPHYCop LFB, alongside
with a metadata indicating the physical port ID that the packet
arrived on.

The LFB is defined with two separate singleton outputs. All Output
packets are emitted in the original Ethernet format received at the
physical port, unchanged, and cover all types of Ethernet types.

The first singleton output is known as "NormalPathOut". It usually
outputs Ethernet packets to some LFB like an EtherClassifier LFB for
further L3 forwarding process alongside with a PHYPortID metadata
indicating which physical port the packet came from.

The second singleton output is known as "L2BridgingPathOut".
Although the LFB library this document defines is basically to meet
typical router functions, it will attempt to be forward compatible
with future router functions. The "L2BridgingPathOut" is defined to
meet the requirement that L2 bridging functions may be optionally
supported simultaneously with L3 processing and some L2 bridging LFBs
that may be defined in the future. If the FE supports L2 bridging,
the CE can enable or disable it by means of a "L2BridgingPathEnable"
component in the FE. If it is enabled, by also instantiating some L2
bridging LFB instances following the L2BridgingPathOut, FEs are
expected to fulfill L2 bridging functions. L2BridgingPathOut will
output packets exactly the same as that in the NormalPathOut output.

This LFB can be set to work in a Promiscuous Mode, allowing all
packets to pass through the LFB without being dropped. Otherwise, a
locality check will be performed based on the local MAC addresses.
All packets that do not pass through the locality check will be
dropped.

This LFB can optionally participate in Ethernet flow control in
cooperation with EtherMACOut LFB. This document does not go into the
details of how this is implemented; the reader may refer to some

relevant references. This document also does not describe how the
buffers which induce the flow control messages behave - it is assumed
that such artifacts exist and describing them is out of scope in this
document.

5.1.2.2. Components

The AdminStatus component is defined for the CE to administratively
manage the status of the LFB. The CE may administratively startup or
shutdown the LFB by changing the value of AdminStatus. The default
value is set to 'Down'.

The LocalMACAddresses component specifies the local MAC addresses
based on which locality checks will be made. This component is an
array of MAC addresses, and of 'read-write' access permission.

An L2BridgingPathEnable component captures whether the LFB is set to
work as a L2 bridge. An FE that does not support bridging will
internally set this flag to false, and additionally set the flag
property as read-only. The default value for is 'false'.

The PromiscuousMode component specifies whether the LFB is set to
work as in a promiscuous mode. The default value for is 'false'.

The TxFlowControl component defines whether the LFB is performing
flow control on sending packets. The default value for is 'false'.
Note that the component is defined as "optional". If an FE does not
implement the component while a CE try to configure the component to
this FE, an error from FE may be responded to CE with error code
like: 0x09(E_COMPONENT_DOES_NOT_EXIST) or 0x15(E_NOT_SUPPORTED)
depending on the FE processing. See [RFC5810] for details.

The RxFlowControl component defines whether the LFB is performing
flow control on receiving packets. The default value for is
'false'.The component is also defined as "optional" one.

A struct component, MACInStats, defines a set of statistics for this
LFB, including the number of received packets and the number of
dropped packets. Note that this statistics component is optional to
implementers. If a CE tries to query the componennt while it is not
implemented in an FE, an error code will be responded to CE
indicating the error type like: 0x09(E_COMPONENT_DOES_NOT_EXIST) or
0x15(E_NOT_SUPPORTED), depending on the FE implementation.

5.1.2.3. Capabilities

This LFB does not have a list of capabilities.

5.1.2.4. Events

   This LFB does not have any events specified.

5.1.3. EtherClassifier

   EtherClassifier LFB abstracts the process to decapsulate Ethernet
   packets and then classify them.

5.1.3.1. Data Handling

   This LFB describes the process of decapsulating Ethernet packets and
   classifying them into various network layer data packets according to
   information included in the Ethernet packets headers.

   The LFB is expected to receive all types of Ethernet packets, via a
   singleton input known as "EtherPktsIn", which are usually output from
   an upstream LFB like EtherMACIn LFB. This input is also capable of
   multiplexing to allow for multiple upstream LFBs being connected.
   For instance, when L2 bridging function is enabled in EtherMACIn LFB,
   some L2 bridging LFBs may be applied. In this case, some Ethernet
   packets after L2 processing may have to be input to EtherClassifier
   LFB for classification, while simultaneously packets directly output
   from EtherMACIn may also need to input to this LFB. This input is
   capable of handling such a case. Usually, all expected Ethernet
   Packets will be associated with a PHYPortID metadata, indicating the
   physical port the packet comes from. In some cases, for instance,
   like in a MACinMAC case, a LogicalPortID metadata may be expected to
   associate with the Ethernet packet to further indicate which logical
   port the Ethernet packet belongs to. Note that PHYPortID metadata is
   always expected while LogicalPortID metadata is optionally expected.

   Two output LFB ports are defined.

   The first output is a group output port known as "ClassifyOut".
   Types of network layer protocol packets are output to instances of
   the port group. Because there may be various type of protocol
   packets at the output ports, the produced output frame is defined as
   arbitrary for the purpose of wide extensibility in the future.
   Metadata to be carried along with the packet data is produced at this
   LFB for consumption by downstream LFBs. The metadata passed
   downstream includes PHYPortID, as well as information on Ethernet
   type, source MAC address, destination MAC address and the logical
   port ID. If the original packet is a VLAN packet and contains a VLAN
   ID and a VLAN priority value, then the VLAN ID and the VLAN priority
   value are also carried downstream as metadata. As a result, the VLAN
   ID and priority metadata are defined with the availability of
   "conditional".

   The second output is a singleton output port known as "ExceptionOut",
   which will output packets for which the data processing failed, along
   with an additional ExceptionID metadata to indicate what caused the
   exception. Currently defined exception types include:

   o There is no matching when classifying the packet.

   Usually the exception out port may point to no where, indicating
   packets with exceptions are dropped, while in some cases, the output
   may be pointed to the path to the CE for further processing,
   depending on individual implementations.

5.1.3.2. Components

   An EtherDispatchTable array component is defined in the LFB to
   dispatch every Ethernet packet to the output group according to the
   logical port ID assigned by the VlanInputTable to the packet and the
   Ethernet type in the Ethernet packet header. Each row of the array
   is a struct containing a Logical Port ID, an EtherType and an Output
   Index. With the CE configuring the dispatch table, the LFB can be
   expected to classify various network layer protocol type packets and
   output them at different output ports. It is expected that the LFB
   classify packets according to protocols like IPv4, IPv6, MPLS, ARP,
   ND, etc.

   A VlanInputTable array component is defined in the LFB to classify
   VLAN Ethernet packets. Each row of the array is a struct containing
   an Incoming Port ID, a VLAN ID and a Logical Port ID. According to
   IEEE VLAN specifications, all Ethernet packets can be recognized as
   VLAN types by defining that if there is no VLAN encapsulation in a
   packet, a case with VLAN tag 0 is considered. Every input packet is
   assigned with a new LogicalPortID according to the packet incoming
   port ID and the VLAN ID. A packet incoming port ID is defined as a
   logical port ID if a logical port ID is associated with the packet,
   or a physical port ID if no logical port ID associated. The VLAN ID
   is exactly the VLAN ID in the packet if it is a VLAN packet, or 0 if
   it is not. Note that a logical port ID of a packet may be rewritten
   with a new one by the VlanInputTable processing.

   Note that the logical port ID and physical port ID mentioned above
   are all originally configured by CE, and are globally effective
   within a ForCES NE (Network Element). To distinguish a physical port
   ID from a logical port ID in the incoming port ID field of the
   VlanInputTable, physical port ID and logical port ID must be assigned
   with separate number spaces.

   An array component, EtherClassifyStats, defines a set of statistics
   for this LFB, measuring the number of packets per EtherType. Each

row of the array is a struct containing an EtherType and a Packet
number.Note that this statistics component is optional to
implementers.

5.1.3.3. Capabilities

This LFB does not have a list of capabilities.

5.1.3.4. Events

This LFB has no events specified.

5.1.4. EtherEncap

The EtherEncap LFB abstracts the process to replace or attach
appropriate Ethernet headers to the packet.

5.1.4.1. Data Handling

This LFB abstracts the process of encapsulating Ethernet headers onto
received packets. The encapsulation is based on passed metadata.

The LFB is expected to receive IPv4 and IPv6 packets, via a singleton
input port known as "EncapIn" which may be connected to an upstream
LFB like an IPv4NextHop, an IPv6NextHop, BasicMetadataDispatch, or
any LFB which requires to output packets for Ethernet encapsulation.
The LFB always expects from upstream LFBs the MediaEncapInfoIndex
metadata which is used as a search key to lookup the encapsulation
table EncapTable by the search key matching the table index. An
input packet may also optionally receive a VLAN priority metadata,
indicating that the packet is originally with a priority value. The
priority value will be loaded back to the packet when encapsulating.
The optional VLAN priority metadata is defined with a default value
0.

Two singleton output LFB ports are defined.

The first singleton output known as "SuccessOut". Upon a successful
table lookup, the destination and source MAC addresses, and the
logical media port (L2PortID) are found in the matching table entry.
The CE may set the VlanID in case VLANs are used. By default the
table entry for VlanID of 0 is used as per IEEE rules. Whatever the
value of VlanID is, if the input metadata VlanPriority is non-zero,
the packet will have a VLAN tag. If the VlanPriority and the VlanID
are all zero, there is no VLAN tag to this packet. After replacing
or attaching the appropriate Ethernet headers to the packet is
complete, the packet is passed out on the "SuccessOut" LFB port to a
downstream LFB instance alongside with the L2PortID.

The second singleton output known as "ExceptionOut", which will
output packets for which the table lookup fails, along with an
additional ExceptionID metadata. Currently defined exception types
only include the following case:

o The MediaEncapInfoIndex value of the packet is invalid and can not
   be allocated in the EncapTable.

o The packet failed lookup of the EncapTable table even though the
   MediaEncapInfoIndex is valid.

The upstream LFB may be programmed by the CE to pass along a
MediaEncapInfoIndex that does not exist in the EncapTable. That is
to allow for resolution of the L2 headers, if needed, to be made at
the L2 encapsulation level in this case (Ethernet) via ARP, or ND (or
other methods depending on the link layer technology) when a table
miss occurs.

For neighbor L2 header resolution(table miss exception), the
processing LFB may pass this packet to the CE via the redirect LFB or
FE software or another LFB instance for further resolution. In such
a case the metadata NextHopIPv4Addr or NextHopIPv6Addr generated by
next hop LFB is also passed to the exception handling. Such an IP
address could be used to do activities such as ARP or ND by the
handler it is passed to.

The result of the L2 resolution is to update the EncapTable as well
as the next hop LFB so subsequent packets do not fail EncapTable
lookup. The EtherEncap LFB does not make any assumptions of how the
EncapTable is updated by the CE (or whether ARP/ND is used
dynamically or static maps exist).

Downstream LFB instances could be either an EtherMACOut type or a
BasicMetadataDispatch type. If the final packet L2 processing is
possible to be on per-media-port basis or resides on a different FE
or in cases where L2 header resolution is needed, then the model
makes sense to use a BasicMetadataDispatch LFB to fan out to
different LFB instances. If there is a direct egress port point,
then the model makes sense to have a downstream LFB instance being an
EtherMACOut.

5.1.4.2. Components

This LFB has only one component named EncapTable which is defined as
an array. Each row of the array is a struct containing the
destination MAC address, the source MAC address, the VLAN ID with a
default value of zero and the output logical L2 port ID.

5.1.4.3. Capabilities

   This LFB does not have a list of capabilities.

5.1.4.4. Events

   This LFB does not have any events specified.

5.1.5. EtherMACOut

   EtherMACOut LFB abstracts an Ethernet port at MAC data link layer.
   This LFB describes Ethernet packet output process. Ethernet output
   functions are closely related to Ethernet input functions, therefore
   many components defined in this LFB are as aliases of EtherMACIn LFB
   components.

5.1.5.1. Data Handling

   The LFB is expected to receive all types of Ethernet packets, via a
   singleton input known as "EtherPktsIn", which are usually output from
   an Ethernet encapsulation LFB, alongside with a metadata indicating
   the physical port ID that the packet will go through.

   The LFB is defined with a singleton output port known as
   "EtherPktsOut". All Output packets are in Ethernet format, possibly
   with various Ethernet types, alongside with a metadata indicating the
   physical port ID the packet is to go through. This output links to a
   downstream LFB that is usually an Ethernet physical LFB like
   EtherPHYcop LFB.

   This LFB can optionally participate in Ethernet flow control in
   cooperation with EtherMACIn LFB. This document does not go into the
   details of how this is implemented; the reader may refer to some
   relevant references. This document also does not describe how the
   buffers which induce the flow control messages behave - it is assumed
   that such artifacts exist and describing them is out of scope in this
   document.

   Note that as a base definition, functions like multiple virtual MAC
   layers are not supported in this LFB version. It may be supported in
   the future by defining a subclass or a new version of this LFB.

5.1.5.2. Components

   The AdminStatus component is defined for CE to administratively
   manage the status of the LFB. The CE may administratively startup or
   shutdown the LFB by changing the value of AdminStatus. The default
   value is set to 'Down'. Note that this component is defined as an

alias of the AdminStatus component in the EtherMACIn LFB. This
infers that an EtherMACOut LFB usually coexists with an EtherMACIn
LFB, both of which share the same administrative status management by
CE. Alias properties as defined in the ForCES FE model [RFC5812]
will be used by CE to declare the target component this alias refers,
which include the target LFB class and instance IDs as well as the
path to the target component.

The MTU component defines the maximum transmission unit.

The optinal TxFlowControl component defines whether the LFB is
performing flow control on sending packets. The default value for is
'false'. Note that this component is defined as an alias of
TxFlowControl component in the EtherMACIn LFB.

The optional RxFlowControl component defines whether the LFB is
performing flow control on receiving packets. The default value for
is 'false'. Note that this component is defined as an alias of
RxFlowControl component in the EtherMACIn LFB.

A struct component, MACOutStats, defines a set of statistics for this
LFB, including the number of transmitted packets and the number of
dropped packets. This statistics component is optional to
impleneters.

5.1.5.3. Capabilities

This LFB does not have a list of capabilities.

5.1.5.4. Events

This LFB does not have any events specified.

5.2. IP Packet Validation LFBs

The LFBs are defined to abstract IP packet validation process. An
IPv4Validator LFB is specifically for IPv4 protocol validation and an
IPv6Validator LFB for IPv6.

5.2.1. IPv4Validator

The IPv4Validator LFB performs IPv4 packets validation.

5.2.1.1. Data Handling

This LFB performs IPv4 validation according to [RFC1812] and its
updates. The IPv4 packet will be output to the corresponding LFB
port the indication whether the packet is unicast, multicast or

whether an exception has occurred or the validation failed.

This LFB always expects, as input, packets which have been indicated as IPv4 packets by an upstream LFB, like an EtherClassifier LFB. There is no specific metadata expected by the input of the LFB.

Four output LFB ports are defined.

All validated IPv4 unicast packets will be output at the singleton port known as "IPv4UnicastOut". All validated IPv4 multicast packets will be output at the singleton port known as "IPv4MulticastOut" port.

A singleton port known as "ExceptionOut" is defined to output packets which have been validated as exception packets. An exception ID metadata is produced to indicate what has caused the exception. An exception case is the case when a packet needs further processing before being normally forwarded. Currently defined exception types include:

o Packet with expired TTL

o Packet with header length more than 5 words

o Packet IP head including Router Alert options

o Packet with exceptional source address

o Packet with exceptional destination address

Note that although TTL is checked in this LFB for validity, operations like TTL decrement are made by the downstream forwarding LFB.

The final singleton port known as "FailOut" is defined for all packets which have errors and failed the validation process. An error case is the case when a packet is unable to be further processed nor forwarded except being dropped. An error ID is associated a packet to indicate the failure reason. Currently defined failure reasons include:

o Packet with size reported less than 20 bytes

o Packet with version is not IPv4

o Packet with header length less than 5 words

o Packet with total length field less than 20 bytes

o Packet with invalid checksum

o Packet with invalid source address

o Packet with invalid destination address

5.2.1.2. **Components**

   This LFB has only one struct component, the
   IPv4ValidatorStatisticsType, which defines a set of statistics for
   validation process, including the number of bad header packets, the
   number of bad total length packets, the number of bad TTL packets,
   and the number of bad checksum packets. This statistics componennt
   is optional to implementers.

5.2.1.3. **Capabilities**

   This LFB does not have a list of capabilities

5.2.1.4. **Events**

   This LFB does not have any events specified.

5.2.2. **IPv6Validator**

   The IPv6Validator LFB performs IPv6 packets validation.

5.2.2.1. **Data Handling**

   This LFB performs IPv6 validation according to [RFC2460] and its
   updates. Then the IPv6 packet will be output to the corresponding
   port regarding of the validation result, whether the packet is a
   unicast or a multicast one, an exception has occurred or the
   validation failed.

   This LFB always expects, as input, packets which have been indicated
   as IPv6 packets by an upstream LFB, like an EtherClassifier LFB.
   There is no specific metadata expected by the input of the LFB.

   Similar to the IPv4validator LFB, IPv6Validator LFB has also defined
   four output ports to emit packets with various validation results.

   All validated IPv6 unicast packets will be output at the singleton
   port known as "IPv6UnicastOut". All validated IPv6 multicast packets
   will be output at the singleton port known as "IPv6MulticastOut"
   port. There is no metadata produced at this LFB.

A singleton port known as "ExceptionOut" is defined to output packets
which have been validated as exception packets. An exception case is
the case when a packet needs further processing before being normally
forwarded. An exception ID metadata is produced to indicate what
caused the exception. Currently defined exception types include:

o Packet with hop limit to zero

o Packet with next header set to Hop-by-Hop

o Packet with exceptional source address

o Packet with exceptional destination address

The final singleton port known as "FailOut" is defined for all
packets which have errors and failed the validation process. An
error case is the case when a packet is unable to be further
processed nor forwarded except being dropped. A validate error ID is
associated to every failed packet to indicate the reason. Currently
defined reasons include:

o Packet with size reported less than 40 bytes

o Packet with not IPv6 version

o Packet with invalid source address

o Packet with invalid destination address

Note that in the base type library, definitions for exception ID and
validate error ID metadata are applied to both IPv4Validator and
IPv6Validator LFBs, i.e., the two LFBs share the same medadata
definition, with different ID assignment inside.

5.2.2.2. Components

This LFB has only one struct component, the
IPv6ValidatorStatisticsType, which defines a set of statistics for
validation process, including the number of bad header packets, the
number of bad total length packets, and the number of bad hop limit
packets. Note that this component is optional to implementers.

5.2.2.3. Capabilities

This LFB does not have a list of capabilities

5.2.2.4. Events

   This LFB does not have any events specified.

5.3. IP Forwarding LFBs

   IP Forwarding LFBs are specifically defined to abstract the IP
   forwarding processes. As definitions for a base LFB library, this
   document restricts its LFB definition scope only to IP unicast
   forwarding. IP multicast may be defined in future documents.

   The two fundamental tasks performed in IP unicast forwarding
   constitute looking up the forwarding information table to find next
   hop information, and then using the resulting next hop details to
   forward packets out on specific physical output ports. This document
   models the forwarding processes by abstracting out the described two
   steps. Whereas this document describes functional LFB models which
   are modular, there may be multiple ways to implement the abstracted
   models. It is not intended nor expected that the provided LFB models
   constrain implementations.

   Based on the IP forwarding abstraction, two kind of typical IP
   unicast forwarding LFBs are defined, Unicast LPM lookup LFB and next
   hop application LFB. They are further distinguished by IPv4 and IPv6
   protocols.

5.3.1. IPv4UcastLPM

   The IPv4UcastLPM LFB abstracts the IPv4 unicast Longest Prefix Match
   (LPM) process.

   This LFB also provides facilities to support users to implement
   equal-cost multi-path routing (ECMP) or reverse path forwarding
   (RPF). However, this LFB itself does not provide ECMP or RPF. To
   fully implement ECMP or RPF, additional specific LFBs, like a
   specific ECMP LFB or an RPF LFB, will have to be defined.

5.3.1.1. Data Handling

   This LFB performs the IPv4 unicast LPM table looking up. It always
   expects as input IPv4 unicast packets from one singleton input known
   as "PktsIn". Then the LFB uses the destination IPv4 address of every
   packet as search key to look up the IPv4 prefix table and generate a
   hop selector as the matching result. The hop selector is passed as
   packet metadata to downstream LFBs, and will usually be used there as
   a search index to find more next hop information.

   Three singleton output LFB ports are defined.

The first singleton output known as "NormalOut" outputs IPv4 unicast
packets that succeed the LPM lookup and (got a hop selector). The
hop selector is associated with the packet as a metadata. Downstream
from the LPM LFB is usually a next hop application LFB, like an
IPv4NextHop LFB.

The second singleton output known as "ECMPOut" is defined to provide
support for users wishing to implement ECMP.

An ECMP flag is defined in the LPM table to enable the LFB to support
ECMP. When a table entry is created with the flag set true, it
indicates this table entry is for ECMP only. A packet, which has
passed through this prefix lookup, will always output from "ECMPOut"
output port, with the hop selector being its lookup result. The
output will usually directly go to a downstream ECMP processing LFB,
where the hop selector can usually further generate optimized one or
multiple next hop routes by use of ECMP algorithms.

A default route flag is defined in the LPM table to enable the LFB to
support a default route as well as loose RPF. When this flag is set
true, the table entry is identified a default route which also
implies that the route is forbidden for RPF. If a user wants to
implement RPF on FE, a specific RPF LFB will have to be defined. In
such RPF LFB, a component can be defined as an alias of the prefix
table component of this LFB as described below.

The final singleton output is known as "ExceptionOut" and is defined
to allow exception packets to output here, along with an ExceptionID
metadata to indicate what caused the exception. Currently defined
exception types include:

o The packet failed the LPM lookup of the prefix table.

The upstream LFB of this LFB is usually IPv4Validator LFB. If RPF is
to be adopted, the upstream can be an RPF LFB, when defined.

The downstream LFB is usually IPv4NextHop LFB. If ECMP is adopted,
the downstream can be an ECMP LFB, when defined.

5.3.1.2. Components

This LFB has two components.

The IPv4PrefixTable component is defined as an array component of the
LFB. Each row of the array contains an IPv4 address, a Prefix
length, a Hop Selector, an ECMP flag and a Default Route flag. The
LFB uses the destination IPv4 address of every input packet as search
key to look up this table in order extract a next hop selector. The

ECMP flag is for the LFB to support ECMP. The default route flag is
for the LFB to support a default route and for loose RPF.

The IPv4UcastLPMStats component is a struct component which collects
statistics information, including the total number of input packets
received, the IPv4 packets forwarded by this LFB and the number of IP
datagrams discarded due to no route found.Note the component is
defined as optional to implementers.

5.3.1.3. Capabilities

This LFB does not have a list of capabilities

5.3.1.4. Events

This LFB does not have any events specified.

5.3.2. IPv4NextHop

This LFB abstracts the process of selecting ipv4 next hop action.

5.3.2.1. Data Handling

The LFB abstracts the process of next hop information application to
IPv4 packets. It receives an IPv4 packet with an associated next hop
identifier (HopSelector), and uses the identifier as a table index to
look up a next hop table to find an appropriate LFB output port.

The LFB is expected to receive unicast IPv4 packets, via a singleton
input known as "PktsIn" along with a HopSelector metadata which is
used as a table index to lookup the NextHop table. The data
processing involves the forwarding TTL decrement and IP checksum
recalculation.

Two output LFB ports are defined.

The first output is a group output port known as "SuccessOut". On
successful data processing the packet is sent out an LFB-port from
within the LFB port group as selected by the LFBOutputSelectIndex
value of the matched table entry. The packet is sent to a downstream
LFB alongside with the L3PortID and MediaEncapInfoIndex metadata.

The second output is a singleton output port known as "ExceptionOut",
which will output packets for which the data processing failed, along
with an additional ExceptionID metadata to indicate what caused the
exception. Currently defined exception types include:

o The HopSelector for the packet is invalid.

o The packet failed lookup of the next hop table even though the
   HopSelector is valid.

o The MTU for outgoing interface is less than the packet size.

   Downstream LFB instances could be either a BasicMetadataDispatch type
   (Section 5.5.1), used to fan out to different LFB instances or a
   media encapsulation related type, such as an EtherEncap type or a
   RedirectOut type(Section 5.4.2). For example, if there are Ethernet
   and other tunnel Encapsulation, then a BasicMetadataDispatch LFB can
   use the L3PortID metadata (Section 5.3.2.2) to dispatch packets to
   different encapsulator.

5.3.2.2. Components

   This LFB has only one component named IPv4NextHopTable which is
   defined as an array. The HopSelector received is used to match the
   array index of IPv4NextHopTable to find out a row of the table as the
   next hop information result. Each row of the array is a struct
   containing:

o The L3PortID, which is the ID of the Logical Output Port that is
   passed onto the downstream LFB instance. This ID indicates what
   port to the neighbor is as defined by L3. Usually this ID is used
   for the next hop LFB to distinguish packets that need different L2
   encapsulating. For instance, some packets may require general
   Ethernet encapsulation while others may require various types of
   tunnel encapsulations. In such case, different L3PortIDs are
   assigned to the packets and are as metadata passed to downstream
   LFB. A BasicMetadataDispatch LFB(Section 5.5.1) may have to be
   applied as the downstream LFB so as to dispatch packets to
   different encapsulation LFB instances according to the L3PortIDs.

o MTU, the Maximum Transmission Unit for the outgoing port.

o NextHopIPAddr, the IPv4 next hop address.

o MediaEncapInfoIndex, the index we pass onto the downstream
   encapsulation LFB instance and that is used there as a search key
   to lookup a table (typically media encapsulation related) for
   further encapsulation information. The search key looks up the
   table by matching the table index.Note that an encapsulation LFB
   instance may not directly follow the next hop LFB, but the index
   is passed as a metadata associated, as such an encapsulation LFB
   instance even further downstream to the next hop LFB can still use
   the index. In some cases, depending on implementation, the CE may

set the MediaEncapInfoIndex passed downstream to a value that will
fail lookup when it gets to a target encapsulation LFB; such a
lookup failure at that point is an indication that further
resolution is needed. For an example of this approach refer to
Section 7.2 which talks about ARP and mentions this approach.

o LFBOutputSelectIndex, the LFB Group output port index to select
   downstream LFB port. It is a 1-to-1 mapping with FEObject LFB's
   table LFBTopology (See [RFC5812]) component FromPortIndex
   corresponding to the port group mapping FromLFBID as IPv4NextHop
   LFB instance.

5.3.2.3. Capabilities

This LFB does not have a list of capabilities

5.3.2.4. Events

This LFB does not have any events specified.

5.3.3. IPv6UcastLPM

The IPv6UcastLPM LFB abstracts the IPv6 unicast Longest Prefix Match
(LPM) process. The definition of this LFB is similar to the
IPv4UcastLPM LFB except that all IP addresses refer to IPv6
addresses.

This LFB also provides facilities to support users to implement
equal-cost multi-path routing (ECMP) or reverse path forwarding
(RPF). However, this LFB itself does not provide ECMP or RPF. To
fully implement ECMP or RPF, additional specific LFBs, like a
specific ECMP LFB or an RPF LFB, will have to be defined. This work
may be done in the future version of the document.

5.3.3.1. Data Handling

This LFB performs the IPv6 unicast LPM table look up. It always
expects as input IPv6 unicast packets from one singleton input known
as "PktsIn". The destination IPv6 address of an incoming packet is
used as search key to look up the IPv6 prefix table and generate a
hop selector. This hop selector result is associated to the packet
as a metadata and sent to downstream LFBs, and will usually be used
in downstream LFBs as a search key to find more next hop information.

Three singleton output LFB ports are defined.

The first singleton output known as "NormalOut" outputs IPv6 unicast
packets that succeed the LPM lookup (and got a hop selector). The

hop selector is associated with the packet as a metadata. Downstream
from the LPM LFB is usually a next hop application LFB, like an
IPv6NextHop LFB.

The second singleton output known as "ECMPOut" is defined to provide
support for users wishing to implement ECMP.

An ECMP flag is defined in the LPM table to enable the LFB to support
ECMP. When a table entry is created with the flag set true, it
indicates this table entry is for ECMP only. A packet, which has
passed through this prefix lookup, will always output from "ECMPOut"
output port, with the hop selector being its lookup result. The
output will usually directly go to a downstream ECMP processing LFB,
where the hop selector can usually further generate optimized one or
multiple next hop routes by use of ECMP algorithms.

A default route flag is defined in the LPM table to enable the LFB to
support a default route as well as loose RPF. When this flag is set
true, the table entry is identified a default route which also
implies that the route is forbidden for RPF.

If a user wants to implement RPF on FE, a specific RPF LFB will have
to be defined. In such RPF LFB, a component can be defined as an
alias of the prefix table component of this LFB as described below.

The final singleton output is known as "ExceptionOut" and is defined
to allow exception packets to output here, along with an ExceptionID
metadata to indicate what caused the exception. Currently defined
exception types include:

o The packet failed the LPM lookup of the prefix table.

The upstream LFB of this LFB is usually IPv6Validator LFB. If RPF is
to be adopted, the upstream can be an RPF LFB, when defined.

The downstream LFB is usually an IPv6NextHop LFB. If ECMP is
adopted, the downstream can be an ECMP LFB, when defined.

5.3.3.2. Components

This LFB has two components.

The IPv6PrefixTable component is defined as an array component of the
LFB. Each row of the array contains an IPv6 address, a Prefix
length, a Hop Selector, an ECMP flag and a Default Route flag. The
ECMP flag is so the LFB can support ECMP. The default route flag is
for the LFB to support a default route and for loose RPF as described
earlier.

The IPv6UcastLPMStats component is a struct component which collects statistics information, including the total number of input packets received, the IPv6 packets forwarded by this LFB and the number of IP datagrams discarded due to no route found.Note the component is defined as optional to implementers.

5.3.3.3. Capabilities

This LFB does not have a list of capabilities

5.3.3.4. Events

This LFB does not have any events specified.

5.3.4. IPv6NextHop

This LFB abstracts the process of selecting IPv6 next hop action.

5.3.4.1. Data Handling

The LFB abstracts the process of next hop information application to IPv6 packets. It receives an IPv6 packet with an associated next hop identifier (HopSelector), and uses the identifier to look up a next hop table to find an appropriate output port from the LFB.

The LFB is expected to receive unicast IPv6 packets, via a singleton input known as "PktsIn" along with a HopSelector metadata which is used as a table index to lookup the next hop table.

Two output LFB ports are defined.

The first output is a group output port known as "SuccessOut". On successful data processing the packet is sent out an LFB port from within the LFB port group as selected by the LFBOutputSelectIndex value of the matched table entry. The packet is sent to a downstream LFB alongside with the L3PortID and MediaEncapInfoIndex metadata.

The second output is a singleton output port known as "ExceptionOut", which will output packets for which the data processing failed, along with an additional ExceptionID metadata to indicate what caused the exception. Currently defined exception types include:

o The HopSelector for the packet is invalid.

o The packet failed lookup of the next hop table even though the HopSelector is valid.

o The MTU for outgoing interface is less than the packet size.

   Downstream LFB instances could be either a BasicMetadataDispatch
   type, used to fan out to different LFB instances or a media
   encapsulatation related type, such as an EtherEncap type or a
   RedirectOut type. For example, when the downstream LFB is
   BasicMetadataDispatch, and there exist Ethernet and other tunnel
   Encapsulation downstream from BasicMetadataDispatch, then the
   BasicMetadataDispatch LFB can use the L3PortID metadata (See section
   below) to dispatch packets to the different encapsulator LFBs.

5.3.4.2. Components

   This LFB has only one component named IPv6NextHopTable which is
   defined as an array. The array index of IPv6NextHopTable is used for
   a HopSelector to find out a row of the table as the next hop
   information. Each row of the array is a struct containing:

o The L3PortID, which is the ID of the Logical Output Port that is
   passed onto the downstream LFB instance. This ID indicates what
   port to the neighbor is as defined by L3. Usually this ID is used
   for the next hop LFB to distinguish packets that need different L2
   encapsulating. For instance, some packets may require general
   Ethernet encapsulation while others may require various types of
   tunnel encapsulations. In such case, different L3PortIDs are
   assigned to the packets and are as metadata passed to downstream
   LFB. A BasicMetadataDispatch LFB(Section 5.5.1) may have to be
   applied as the downstream LFB so as to dispatch packets to
   different encapsulation LFB instances according to the L3PortIDs.

o MTU, the Maximum Transmission Unit for the outgoing port.

o NextHopIPAddr, the IPv6 next hop address.

o MediaEncapInfoIndex, the index we pass onto the downstream
   encapsulation LFB instance and that is used there as a search key
   to lookup a table (typically media encapsulation related) for
   further encapsulation information. The saearch key looks up the
   table by matching the table index. Note that an encapsulation LFB
   instance may not directly follow the next hop LFB, but the index
   is passed as a metadata associated, as such an encapsulation LFB
   instance even further downstream to the next hop LFB can still use
   the index. In some cases, depending on implementation, the CE may
   set the MediaEncapInfoIndex passed downstream to a value that will
   fail lookup when it gets to a target encapsulation LFB; such a
   lookup failure at that point is an indication that further
   resolution is needed. For an example of this approach refer to
   Section 7.2 which talks about ARP and mentions this approach.

   o LFBOutputSelectIndex, the LFB Group output port index to select
      downstream LFB port. It is a 1-to-1 mapping with FEObject LFB's
      table LFBTopology (See [RFC5812]) component FromPortIndex
      corresponding to the port group mapping FromLFBID as IPv4NextHop
      LFB instance.

5.3.4.3. Capabilities

   This LFB does not have a list of capabilities

5.3.4.4. Events

   This LFB does not have any events specified.

5.4. Redirect LFBs

   Redirect LFBs abstract data packets transportation process between CE
   and FE. Some packets output from some LFBs may have to be delivered
   to CE for further processing, and some packets generated by CE may
   have to be delivered to FE and further to some specific LFBs for data
   path processing. According to [RFC5810], data packets and their
   associated metadata are encapsulated in ForCES redirect message for
   transportation between CE and FE. We define two LFBs to abstract the
   process, a RedirectIn LFB and a RedirectOut LFB. Usually, in an LFB
   topology of an FE, only one RedirectIn LFB instance and one
   RedirectOut LFB instance exist.

5.4.1. RedirectIn

   RedirectIn LFB abstracts the process for the CE to inject data
   packets into the FE data path.

5.4.1.1. Data Handling

   A RedirectIn LFB abstracts the process for the CE to inject data
   packets into the FE LFB topology so as to input data packets into FE
   data paths. From LFB topology point of view, the RedirectIn LFB acts
   as a source point for data packets coming from CE, therefore
   RedirectIn LFB is defined with a single output LFB port (and no input
   LFB port).

   The single output port of RedirectIn LFB is defined as a group output
   type, with the name of "PktsOut". Packets produced by this output
   will have arbitrary frame types decided by the CE which generated the
   packets. Possible frames may include IPv4, IPv6, or ARP protocol
   packets. The CE may associate some metadata to indicate the frame
   types and may also associate other metadata to indicate various
   information on the packets. Among them, there MUST exist a

'RedirectIndex' metadata, which is an integer acting as an index.
When the CE transmits the metadata along with the packet to a
RedirectIn LFB, the LFB will read the RedirectIndex metadata and
output the packet to one of its group output port instance, whose
port index is indicated by this metadata. Any other metadata, in
addition to 'RedirectIndex', will be passed untouched along the
packet delivered by the CE to downstream LFB. This means the
'RedirectIndex' metadata from CE will be "consumed" by the RedirectIn
LFB and will not be passed to downstream LFB. Note that, a packet
from CE without a 'RedirectIndex' metadata associated will be dropped
by the LFB. Note that all metadata visible to the LFB need to be
global and IANA controlled. See the "IANA Considerations" of the
document for more details, where a metadata ID private space that can
be used by vendors is also provided.

5.4.1.2. Components

An optional statistics component is defined to collect the number of
packets received by the LFB from CE. There are no other components
defined for current version of the LFB.

5.4.1.3. Capabilities

This LFB does not have a list of capabilities

5.4.1.4. Events

This LFB does not have any events specified.

5.4.2. RedirectOut

RedirectOut LFB abstracts the process for LFBs in the FE to deliver
data packets to the CE.

5.4.2.1. Data Handling

A RedirectOut LFB abstracts the process for LFBs in the FE to deliver
data packets to the CE. From the LFB's topology point of view, the
RedirectOut LFB acts as a sink point for data packets going to the
CE, therefore RedirectOut LFB is defined with a single input LFB port
(and no output LFB port).

The RedirectOut LFB has only one singleton input known as "PktsIn",
but is capable of receiving packets from multiple LFBs by
multiplexing this input. The input expects any kind of frame type
therefore the frame type has been specified as arbitrary, and also
all types of metadata are expected. All associated metadata produced
(but not consumed) by previous processed LFBs should be delivered to

CE via the ForCES protocol redirect message [RFC5810]. The CE can
decide on how to process the redirected packet by referencing the
associated metadata. As an example, a packet could be redirected by
the FE to the CE because the EtherEncap LFB is not able to resolve L2
information. The metadata "ExceptionID", created by the EtherEncap
LFB is passed along with the packet and should be sufficient for the
CE to do the necessary processing and resolve the L2 entry required.
Note that all metadata visible to the LFB need to be global and IANA
controlled. See the "IANA Considerations" of the document for more
details, where a metadata ID private space that can be used by
vendors is also provided.

5.4.2.2. Components

An optional statistics component is defined to collect the number of
packets sent by the LFB to CE. There are no other components defined
for current version of the LFB.

5.4.2.3. Capabilities

This LFB does not have a list of capabilities

5.4.2.4. Events

This LFB does not have any events specified.

5.5. General Purpose LFBs

5.5.1. BasicMetadataDispatch

The BasicMetadataDispatch LFB is defined to abstract the process in
which a packet is dispatched to some output path based on its
associated metadata value.

5.5.1.1. Data Handling

The BasicMetadataDispatch has only one singleton input known as
"PktsIn". Every input packet should be associated with a metadata
that will be used by the LFB to do the dispatch. This LFB contains a
metadata ID and a dispatch table named MetadataDispatchTable, all
configured by the CE. The metadata ID specifies which metadata is to
be used for dispatching packets. The MetadataDispatchTable contains
entries of a metadata value and an OutputIndex, specifying that the
packet with the metadata value must go out from the LFB group output
port instance with the OutputIndex.

Two output LFB ports are defined.

The first output is a group output port known as "PktsOut". A packet
with its associated metadata having found an OutputIndex by
successfully looking up the dispatch table will be output to the
group port instance with the corresponding index.

The second output is a singleton output port known as "ExceptionOut",
which will output packets for which the data processing failed, along
with an additional ExceptionID metadata to indicate what caused the
exception. Currently defined exception types only include one case:

o There is no matching when looking up the metadata dispatch table.

As an example, if the CE decides to dispatch packets according to a
physical port ID (PHYPortID), the CE may set the ID of PHYPortID
metadata to the LFB first. Moreover, the CE also sets the PHYPortID
actual values (the metadata values) and assigned OutputIndex for the
values to the dispatch table in the LFB. When a packet arrives, a
PHYPortID metadata is found associated with the packet, the metadata
value is further used as a key to look up the dispatch table to find
out an output port instance for the packet.

Currently the BasicMetadataDispatch LFB only allows the metadata
value of the dispatch table entry be 32-bits integer. A metadata
with other types of value is not supported in this version. A more
complex metadata dispatch LFB may be defined in future version of the
library. In that LFB, multiple tuples of metadata with more value
types supported may be used to dispatch packets.

5.5.1.2. Components

This LFB has two components. One component is MetadataID and the
other is MetadataDispatchTable. Each row entry of the dispatch table
is a struct containing metadata value and the OutputIndex. Note that
currently, the metadata value is only allowed to be 32-bits integer.
The metadata value is also defined as a content key for the table.
The concept of content key is a searching key for tables which is
defined in the ForCES FE Model [RFC5812]. With the content key, CE
can manipulate the table by means of a specific metadata value rather
than by the table index only. See [RFC5812] document and also the
ForCES Protocol [RFC5810] for more details on the definition and use
of a content key.

5.5.1.3. Capabilities

This LFB does not have a list of capabilities

5.5.1.4. **Events**

   This LFB does not have any events specified.

5.5.2. **GenericScheduler**

   This is a preliminary generic scheduler LFB for abstracting a simple
   scheduling process.

5.5.2.1. **Data Handling**

   There exist various kinds of scheduling strategies with various
   implementations. As a base LFB library, this document only defines a
   preliminary generic scheduler LFB for abstracting a simple scheduling
   process. Users may use this LFB as a basic LFB to further construct
   more complex scheduler LFBs by means of "inheritance" as described in
   [RFC5812].

   Packets of any arbitrary frame type are received via a group input
   known as "PktsIn" with no additional metadata expected. This group
   input is capable of multiple input port instances. Each port
   instance may be connected to different upstream LFB output. Inside
   the LFB, it is abstracted that each input port instance is connected
   to a queue, and the queue is marked with a queue ID whose value is
   exactly the same as the index of corresponding group input port
   instance. Scheduling disciplines are applied to all queues and also
   all packets in the queues.The group input port property
   PortGroupLimits in ObjectLFB as defined by ForCES FE model[RFC5810]
   provides means for the CE to query the capability of total queue
   numbers the scheduler supports. The CE can then decide how many
   queues it may use for a scheduling application.

   Scheduled packets are output from a singleton output port of the LFB
   knows as "PktsOut" with no corresponding metadata.

   More complex scheduler LFBs may be defined with more complex
   scheduling disciplines by succeeding this LFB. For instance, a
   priority scheduler LFB may be defined by inheriting this LFB and
   defining a component to indicate priorities for all input queues.

5.5.2.2. **Components**

   The SchedulingDiscipline component is for the CE to specify a
   scheduling discipline to the LFB. Currently defined scheduling
   disciplines only include Round Robin (RR) strategy. The default
   scheduling discipline is RR then.

   The QueueStats component is defined to allow CE to query every queue

status of the scheduler. It is an array component and each row of
the array is a struct containing a queue ID. Currently defined queue
status includes the queue depth in packets and the queue depth in
bytes. Using the queue ID as the index, the CE can query every queue
for its used length in unit of packets or bytes.Note that the
QueueStats componennt is defined as optional to implementers.

5.5.2.3. Capabilities

The following capability is currently defined for the
GenericScheduler.

o   The queue length limit providing the storage ability for every
    queue.

5.5.2.4. Events

This LFB does not have any events specified.

6. XML for LFB Library

```
<?xml version="1.0" encoding="UTF-8"?>
<LFBLibrary xmlns="urn:ietf:params:xml:ns:forces:lfbmodel:1.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      provides="BaseLFBLibrary">
   <load library="BaseTypeLibrary"/>
   <LFBClassDefs>
      <LFBClassDef LFBClassID="3">
         <name>EtherPHYCop</name>
         <synopsis>
           The EtherPHYCop LFB describes an Ethernet interface
           which limits the physical media to copper.
         </synopsis>
         <version>1.0</version>
         <inputPorts>
            <inputPort>
               <name>EtherPHYIn</name>
               <synopsis>
                 The input port of the EtherPHYCop LFB. It expects any
                 type of Ethernet frame.
               </synopsis>
               <expectation>
                  <frameExpected>
                     <ref>EthernetAll</ref>
                  </frameExpected>
               </expectation>
            </inputPort>
         </inputPorts>
         <outputPorts>
            <outputPort>
               <name>EtherPHYOut</name>
               <synopsis>
                 The output port of the EtherPHYCop LFB. The output
                 packet has the same Ethernet frame type with the
                 input packet, associated with a metadata indicating
                 the ID of the physical port.
               </synopsis>
               <product>
                  <frameProduced>
                     <ref>EthernetAll</ref>
                  </frameProduced>
                  <metadataProduced>
                     <ref>PHYPortID</ref>
                  </metadataProduced>
               </product>
            </outputPort>
```

```
            </outputPorts>
            <components>
               <component componentID="1" access="read-only">
                  <name>PHYPortID</name>
                  <synopsis>
                    The identification of the physical port
                  </synopsis>
                  <typeRef>uint32</typeRef>
               </component>
               <component componentID="2" access="read-write">
                  <name>AdminStatus</name>
                  <synopsis>
                    The port status administratively requested
                  </synopsis>
                  <typeRef>PortStatusType</typeRef>
                  <defaultValue>2</defaultValue>
               </component>
               <component componentID="3" access="read-only">
                  <name>OperStatus</name>
                  <synopsis>
                    The port actual operational status
                  </synopsis>
                  <typeRef>PortStatusType</typeRef>
               </component>
               <component componentID="4" access="read-write">
                  <name>AdminLinkSpeed</name>
                  <synopsis>
                    The port link speed administratively requested
                  </synopsis>
                  <typeRef>LANSpeedType</typeRef>
                  <defaultValue>LAN_SPEED_AUTO</defaultValue>
               </component>
               <component componentID="5" access="read-only">
                  <name>OperLinkSpeed</name>
                  <synopsis>
                    The port actual operational link speed
                  </synopsis>
                  <typeRef>LANSpeedType</typeRef>
               </component>
               <component componentID="6" access="read-write">
                  <name>AdminDuplexMode</name>
                  <synopsis>
                    The port duplex mode administratively requested
                  </synopsis>
                  <typeRef>DuplexType</typeRef>
                  <defaultValue>Auto</defaultValue>
               </component>
               <component componentID="7" access="read-only">
```

```
            <name>OperDuplexMode</name>
            <synopsis>
              The port actual operational duplex mode
            </synopsis>
            <typeRef>DuplexType</typeRef>
        </component>
        <component componentID="8" access="read-only">
            <name>CarrierStatus</name>
            <synopsis>The carrier status of the port </synopsis>
            <typeRef>boolean</typeRef>
            <defaultValue>false</defaultValue>
        </component>
    </components>
    <capabilities>
        <capability componentID="30">
            <name>SupportedLinkSpeed</name>
            <synopsis>
              A list of link speeds the port supports
            </synopsis>
            <array>
                <typeRef>LANSpeedType</typeRef>
            </array>
        </capability>
        <capability componentID="31">
            <name>SupportedDuplexMode</name>
            <synopsis>
              A list of duplex modes the port supports
            </synopsis>
            <array>
                <typeRef>DuplexType</typeRef>
            </array>
        </capability>
    </capabilities>
    <events baseID="60">
        <event eventID="1">
            <name>PHYPortStatusChanged</name>
            <synopsis>
              An event reporting change on operational status of the
              physical port.
            </synopsis>
            <eventTarget>
                <eventField>OperStatus</eventField>
            </eventTarget>
            <eventChanged/>
            <eventReports>
                <eventReport>
                    <eventField>OperStatus</eventField>
                </eventReport>
```

```
                 </eventReports>
              </event>
              <event eventID="2">
                 <name>LinkSpeedChanged</name>
                 <synopsis>
                   An event reporting change on operational link speed
                   of the physical port.
                 </synopsis>
                 <eventTarget>
                    <eventField>OperLinkSpeed</eventField>
                 </eventTarget>
                 <eventChanged/>
                 <eventReports>
                    <eventReport>
                       <eventField>OperLinkSpeed</eventField>
                    </eventReport>
                 </eventReports>
              </event>
              <event eventID="3">
                 <name>DuplexModeChanged</name>
                 <synopsis>
                   An event reporting change on operational duplex mode
                   of the physical port.
                 </synopsis>
                 <eventTarget>
                    <eventField>OperDuplexMode</eventField>
                 </eventTarget>
                 <eventChanged/>
                 <eventReports>
                    <eventReport>
                       <eventField>OperDuplexMode</eventField>
                    </eventReport>
                 </eventReports>
              </event>
           </events>
       </LFBClassDef>
       <LFBClassDef LFBClassID="4">
          <name>EtherMACIn</name>
          <synopsis>
            EtherMACIn LFB describes an Ethernet port at MAC data link
            layer. The LFB describes Ethernet processing functions
            of MAC address locality check, deciding if the Ethernet
            packets should be bridged, providing Ethernet layer flow
            control, etc.
          </synopsis>
          <version>1.0</version>
          <inputPorts>
            <inputPort group="false">
```

```
              <name>EtherPktsIn</name>
              <synopsis>
                The input port of the EtherMACIn LFB. It expects any
                type of Ethernet frame.
              </synopsis>
              <expectation>
                 <frameExpected>
                    <ref>EthernetAll</ref>
                 </frameExpected>
                 <metadataExpected>
                    <ref>PHYPortID</ref>
                 </metadataExpected>
              </expectation>
           </inputPort>
        </inputPorts>
        <outputPorts>
           <outputPort group="false">
              <name>NormalPathOut</name>
              <synopsis>
                An output port in the EtherMACIn LFB. It outputs
                Ethernet packets to downstream LFBs for normal
                processing like Ethernet packet classification and
                other L3 IP layer processing.
              </synopsis>
              <product>
                 <frameProduced>
                    <ref>EthernetAll</ref>
                 </frameProduced>
                 <metadataProduced>
                    <ref>PHYPortID</ref>
                 </metadataProduced>
              </product>
           </outputPort>
           <outputPort>
              <name>L2BridgingPathOut</name>
              <synopsis>
                An output port in
                the EtherMACIn LFB. It outputs Ethernet packets
                to downstream LFBs for layer 2 bridging processing.
                The port is switched on or off by the
                L2BridgingPathEnable flag in the LFB.
              </synopsis>
              <product>
                 <frameProduced>
                    <ref>EthernetAll</ref>
                 </frameProduced>
                 <metadataProduced>
                    <ref>PHYPortID</ref>
```

```
                    </metadataProduced>
                </product>
            </outputPort>
        </outputPorts>
        <components>
            <component componentID="1" access="read-write">
                <name>AdminStatus</name>
                <synopsis>
                    The LFB status administratively requested, which has
                    the same data type with a port status. Default is in
                    'down' status.
                </synopsis>
                <typeRef>PortStatusType</typeRef>
                <defaultValue>2</defaultValue>
            </component>
            <component componentID="2" access="read-write">
                <name>LocalMACAddresses</name>
                <synopsis>
                  Local MAC address(es) of the Ethernet port the LFB
                  represents.
                </synopsis>
                <array>
                    <typeRef>IEEEMAC</typeRef>
                </array>
            </component>
            <component componentID="3" access="read-write">
                <name>L2BridgingPathEnable</name>
                <synopsis>
                  A flag indicating if the LFB L2 BridgingPath output
                  port is enabled or not. Default is not enabled.
                </synopsis>
                <typeRef>boolean</typeRef>
                <defaultValue>false</defaultValue>
            </component>
            <component componentID="4" access="read-write">
                <name>PromiscuousMode</name>
                <synopsis>
                  A flag indicating whether the LFB is in promiscuous
                  mode or not. Default is not.
                </synopsis>
                <typeRef>boolean</typeRef>
                <defaultValue>false</defaultValue>
            </component>
            <component componentID="5" access="read-write">
                <name>TxFlowControl</name>
                <synopsis>
                  A flag indicating whether transmit flow control is
                  applied or not. Default is not.
```

```
              </synopsis>
              <optional/>
              <typeRef>boolean</typeRef>
              <defaultValue>false</defaultValue>
          </component>
          <component componentID="6" access="read-write">
              <name>RxFlowControl</name>
              <synopsis>
                A flag indicating whether receive flow control is
                applied or not. Default is not.
              </synopsis>
              <optional/>
              <typeRef>boolean</typeRef>
              <defaultValue>false</defaultValue>
          </component>
          <component componentID="7" access="read-reset">
              <name>MACInStats</name>
              <synopsis>
                The statistics of the EtherMACIn LFB
              </synopsis>
              <optional/>
              <typeRef>MACInStatsType</typeRef>
          </component>
        </components>
    </LFBClassDef>
    <LFBClassDef LFBClassID="5">
        <name>EtherClassifier</name>
        <synopsis>
          EtherClassifier LFB describes the process to decapsulate
          Ethernet packets and then classify them into various
          network layer packets according to information in the
          Ethernet headers. It is expected the LFB classifies packets
          by packet types like IPv4, IPv6, MPLS, ARP, ND, etc.
        </synopsis>
        <version>1.0</version>
        <inputPorts>
          <inputPort>
            <name>EtherPktsIn</name>
            <synopsis>
              Input port of Ethernet packets. PHYPortID metadata is
              always expected while LogicalPortID metadata is
              optionally expected to associate with every input
              Ethernet packet.
            </synopsis>
            <expectation>
              <frameExpected>
                <ref>EthernetAll</ref>
              </frameExpected>
```

```
                  <metadataExpected>
                      <ref>PHYPortID</ref>
                      <ref dependency="optional" defaultValue="0">
                  LogicalPortID</ref>
                  </metadataExpected>
              </expectation>
          </inputPort>
      </inputPorts>
      <outputPorts>
          <outputPort group="true">
              <name>ClassifyOut</name>
              <synopsis>
                A group port for output of Ethernet classifying
                results.
              </synopsis>
              <product>
                  <frameProduced>
                      <ref>Arbitrary</ref>
                  </frameProduced>
                  <metadataProduced>
                      <ref>PHYPortID</ref>
                      <ref>SrcMAC</ref>
                      <ref>DstMAC</ref>
                      <ref>EtherType</ref>
                      <ref availability="conditional">VlanID</ref>
                      <ref availability="conditional">VlanPriority</ref>
                  </metadataProduced>
              </product>
          </outputPort>
          <outputPort group="false">
              <name>ExceptionOut</name>
              <synopsis>
                A singleton port for output of all Ethernet packets
                that fail the classifying process. An ExceptionID
                metadata indicates the failure reason.
              </synopsis>
              <product>
                  <frameProduced>
                      <ref>Arbitrary</ref>
                  </frameProduced>
                  <metadataProduced>
                      <ref>ExceptionID</ref>
                  </metadataProduced>
              </product>
          </outputPort>
      </outputPorts>
      <components>
          <component access="read-write" componentID="1">
```

```
            <name>EtherDispatchTable</name>
            <synopsis>
              An EtherDispatchTable array component which is defined
              in the LFB to dispatch every Ethernet packet to output
              ports according to logical port ID assigned by the
              VlanInputTable in the LFB and Ethernet type in the
              Ethernet packet header.
            </synopsis>
            <typeRef>EtherDispatchTableType</typeRef>
          </component>
          <component access="read-write" componentID="2">
            <name>VlanInputTable</name>
            <synopsis>
              A VlanInputTable array component which is defined in
              the LFB to classify VLAN Ethernet packets. Every input
              packet is assigned with a new LogicalPortID according
              to the packet incoming port ID and VLAN ID.
            </synopsis>
            <typeRef>VlanInputTableType</typeRef>
          </component>
          <component access="read-reset" componentID="3">
            <name>EtherClassifyStats</name>
            <synopsis>
              A table recording statistics on the Ethernet
              classifying process in the LFB.
            </synopsis>
            <optional/>
            <typeRef>EtherClassifyStatsTableType</typeRef>
          </component>
        </components>
      </LFBClassDef>
      <LFBClassDef LFBClassID="6">
        <name>EtherEncap</name>
        <synopsis>
          The EtherEncap LFB abstracts the process of encapsulating
          Ethernet headers onto received packets. The encapsulation
          is based on passed metadata.
        </synopsis>
        <version>1.0</version>
        <inputPorts>
          <inputPort group="false">
            <name>EncapIn</name>
            <synopsis>
              A input port receiving IPv4 and/or IPv6 packets for
              encapsulation. A MediaEncapInfoIndex metadata is
              expected and a VLAN priority metadata is optionally
              expected with every input packet.
            </synopsis>
```

```
                <expectation>
                <frameExpected>
                   <ref>IPv4</ref>
                   <ref>IPv6</ref>
                </frameExpected>
                <metadataExpected>
                   <ref>MediaEncapInfoIndex</ref>
                   <ref dependency="optional" defaultValue="0">
                   VlanPriority</ref>
                </metadataExpected>
                </expectation>
            </inputPort>
        </inputPorts>
        <outputPorts>
            <outputPort group="false">
                <name>SuccessOut</name>
                <synopsis>
                  An output port for packets which have found Ethernet
                  L2 information and have been successfully encapsulated
                  into an Ethernet packet. A L2portID metadata is
                  produced for every output packet.
                </synopsis>
                <product>
                   <frameProduced>
                      <ref>IPv4</ref>
                      <ref>IPv6</ref>
                   </frameProduced>
                   <metadataProduced>
                      <ref>L2PortID</ref>
                   </metadataProduced>
                </product>
            </outputPort>
            <outputPort group="false">
                <name>ExceptionOut</name>
                <synopsis>
                  An output port for packets that fail encapsulation
                  in the LFB. An ExceptionID metadata indicates failure
                  reason.
                </synopsis>
                <product>
                   <frameProduced>
                      <ref>IPv4</ref>
                      <ref>IPv6</ref>
                   </frameProduced>
                   <metadataProduced>
                      <ref>ExceptionID</ref>
                      <ref>MediaEncapInfoIndex</ref>
                      <ref availability="conditional">VlanPriority</ref>
```

```
                    </metadataProduced>
                 </product>
              </outputPort>
           </outputPorts>
           <components>
              <component componentID="1" access="read-write">
                 <name>EncapTable</name>
                 <synopsis>
                   An array table for Ethernet encapsulation information
                   lookup. Each row of the array contains destination MAC
                   address, source MAC address, VLAN ID, and output
                   logical L2 port ID.
                 </synopsis>
                 <typeRef>EncapTableType</typeRef>
              </component>
           </components>
        </LFBClassDef>
        <LFBClassDef LFBClassID="7">
           <name>EtherMACOut</name>
           <synopsis>
             EtherMACOut LFB abstracts an Ethernet port at MAC data link
             layer. It specifically describes Ethernet packet process
             for output to physical port. A downstream LFB is usually an
             Ethernet physical LFB like EtherPHYcop LFB. Note that
             Ethernet output functions are closely related to Ethernet
             input functions, therefore some components defined in this
             LFB are as alias of EtherMACIn LFB.
           </synopsis>
           <version>1.0</version>
           <inputPorts>
              <inputPort group="false">
                 <name>EtherPktsIn</name>
                 <synopsis>
                   The input port of the EtherMACOut LFB. It expects any
                   type of Ethernet frame.
                 </synopsis>
                 <expectation>
                    <frameExpected>
                       <ref>EthernetAll</ref>
                    </frameExpected>
                    <metadataExpected>
                       <ref>PHYPortID</ref>
                    </metadataExpected>
                 </expectation>
              </inputPort>
           </inputPorts>
           <outputPorts>
              <outputPort group="false">
```

```
        <name>EtherPktsOut</name>
        <synopsis>
          A port to output all Ethernet packets, each with a
          metadata indicating the physical port ID the packet
          is to go.
        </synopsis>
        <product>
           <frameProduced>
              <ref>EthernetAll</ref>
           </frameProduced>
           <metadataProduced>
              <ref>PHYPortID</ref>
           </metadataProduced>
        </product>
      </outputPort>
    </outputPorts>
    <components>
       <component componentID="1" access="read-write">
          <name>AdminStatus</name>
          <synopsis>
            The LFB status administratively requested, which has
            the same data type with a port status. The component
            is defined as alias of AdminStatus component in
            EtherMACIn LFB.
          </synopsis>
          <alias>PortStatusType</alias>
       </component>
       <component componentID="2" access="read-write">
          <name>MTU</name>
          <synopsis>Maximum transmission unit (MTU) </synopsis>
          <typeRef>uint32</typeRef>
       </component>
       <component componentID="3" access="read-write">
          <name>TxFlowControl</name>
          <synopsis>
            A flag indicating whether transmit flow control is
            applied, defined as alias of TxFlowControl component
            in EtherMACIn LFB.
          </synopsis>
          <optional/>
          <alias>boolean</alias>
       </component>
       <component componentID="4" access="read-write">
          <name>RxFlowControl</name>
          <synopsis>
            A flag indicating whether receive flow control is
            applied, defined as alias of RxFlowControl component
            in EtherMACIn LFB.
```

```
                     </synopsis>
                     <optional/>
                     <alias>boolean</alias>
                 </component>
                 <component componentID="5" access="read-reset">
                     <name>MACOutStats</name>
                     <synopsis>
                        The statistics of the EtherMACOut LFB
                     </synopsis>
                     <optional/>
                     <typeRef>MACOutStatsType</typeRef>
                 </component>
             </components>
        </LFBClassDef>
        <LFBClassDef LFBClassID="8">
            <name>IPv4Validator</name>
            <synopsis>
              This LFB performs IPv4 validation according to RFC1812 and
              its updates. The IPv4 packet will be output to the
              corresponding LFB port, indicating whether the packet is
              unicast, multicast or whether an exception has occurred or
              the validation failed.
            </synopsis>
            <version>1.0</version>
            <inputPorts>
               <inputPort>
                  <name>ValidatePktsIn</name>
                  <synopsis>
                     Input port for data packets to be validated
                  </synopsis>
                  <expectation>
                      <frameExpected>
                         <ref>Arbitrary</ref>
                      </frameExpected>
                  </expectation>
               </inputPort>
            </inputPorts>
            <outputPorts>
               <outputPort>
                  <name>IPv4UnicastOut</name>
                  <synopsis>
                     Output port for validated IPv4 unicast packets
                  </synopsis>
                  <product>
                      <frameProduced>
                         <ref>IPv4Unicast</ref>
                      </frameProduced>
                  </product>
```

```
            </outputPort>
            <outputPort>
               <name>IPv4MulticastOut</name>
               <synopsis>
                 Output port for validated IPv4 multicast packets
               </synopsis>
               <product>
                  <frameProduced>
                     <ref>IPv4Multicast</ref>
                  </frameProduced>
               </product>
            </outputPort>
            <outputPort>
               <name>ExceptionOut</name>
               <synopsis>
                 Output port for all packets with exceptional cases
                 when validating. An ExceptionID metadata indicates
                 the exception case type.
               </synopsis>
               <product>
                  <frameProduced>
                     <ref>IPv4</ref>
                  </frameProduced>
                  <metadataProduced>
                     <ref>ExceptionID</ref>
                  </metadataProduced>
               </product>
            </outputPort>
            <outputPort>
               <name>FailOut</name>
               <synopsis>
                 Output port for packets failed validating process.
                 A ValidateErrorID metadata indicates the error type
                 or failure reason.
               </synopsis>
               <product>
                  <frameProduced>
                     <ref>IPv4</ref>
                  </frameProduced>
                  <metadataProduced>
                     <ref>ValidateErrorID</ref>
                  </metadataProduced>
               </product>
            </outputPort>
         </outputPorts>
         <components>
            <component access="read-write" componentID="1">
               <name>IPv4ValidatorStats</name>
```

```
              <synopsis>
                The statistics information for validating process in
                the LFB.
              </synopsis>
              <optional/>
              <typeRef>IPv4ValidatorStatsType</typeRef>
          </component>
        </components>
     </LFBClassDef>
      <LFBClassDef LFBClassID="9">
         <name>IPv6Validator</name>
         <synopsis>
           This LFB performs IPv6 validation according to RFC2460 and
           its updates. Then the IPv6 packet will be output to the
           corresponding port, indicating whether the packet is a
           unicast or a multicast one, an exception has occurred or
           the validation failed.
         </synopsis>
         <version>1.0</version>
         <inputPorts>
            <inputPort>
               <name>ValidatePktsIn</name>
               <synopsis>
                 Input port for data packets to be validated
               </synopsis>
               <expectation>
                  <frameExpected>
                     <ref>Arbitrary</ref>
                  </frameExpected>
               </expectation>
            </inputPort>
         </inputPorts>
         <outputPorts>
            <outputPort>
               <name>IPv6UnicastOut</name>
               <synopsis>
                 Output port for validated IPv6 unicast packets
               </synopsis>
               <product>
                  <frameProduced>
                     <ref>IPv6Unicast</ref>
                  </frameProduced>
               </product>
            </outputPort>
            <outputPort>
               <name>IPv6MulticastOut</name>
               <synopsis>
                 Output port for validated IPv6 multicast packets
```

```
                        </synopsis>
                        <product>
                           <frameProduced>
                              <ref>IPv6Multicast</ref>
                           </frameProduced>
                        </product>
                     </outputPort>
                     <outputPort>
                        <name>ExceptionOut</name>
                        <synopsis>
                          Output port for packets with exceptional cases when
                          validating. An ExceptionID metadata indicates the
                          exception case type.
                        </synopsis>
                        <product>
                           <frameProduced>
                              <ref>IPv6</ref>
                           </frameProduced>
                           <metadataProduced>
                              <ref>ExceptionID</ref>
                           </metadataProduced>
                        </product>
                     </outputPort>
                     <outputPort>
                        <name>FailOut</name>
                        <synopsis>
                          Output port for packets failed validating process.
                          A ValidateErrorID metadata indicates the error type
                          or failure reason.
                        </synopsis>
                        <product>
                           <frameProduced>
                              <ref>IPv6</ref>
                           </frameProduced>
                           <metadataProduced>
                              <ref>ValidateErrorID</ref>
                           </metadataProduced>
                        </product>
                     </outputPort>
                  </outputPorts>
                  <components>
                     <component access="read-write" componentID="1">
                        <name>IPv6ValidatorStats</name>
                        <synopsis>
                          The statistics information for validating process in
                          the LFB.
                        </synopsis>
                        <optional/>
```

```
                  <typeRef>IPv6ValidatorStatsType</typeRef>
                </component>
            </components>
        </LFBClassDef>
      <LFBClassDef LFBClassID="10">
          <name>IPv4UcastLPM</name>
          <synopsis>
            The IPv4UcastLPM LFB abstracts the IPv4 unicast Longest
            Prefix Match (LPM) process. This LFB supports to implement
            equal-cost multi-path routing (ECMP) and reverse path
            forwarding (RPF).
          </synopsis>
          <version>1.0</version>
          <inputPorts>
              <inputPort group="false">
                  <name>PktsIn</name>
                  <synopsis>
                    A port for input of packets to be processed. IPv4
                    unicast packets are expected.
                  </synopsis>
                  <expectation>
                  <frameExpected>
                      <ref>IPv4Unicast</ref>
                  </frameExpected>
                  </expectation>
              </inputPort>
          </inputPorts>
          <outputPorts>
              <outputPort group="false">
                  <name>NormalOut</name>
                  <synopsis>
                    An output port to output IPv4 unicast packets
                    successfully passed the LPM lookup. A HopSelector
                    metadata is produced to associate every output packet
                    for downstream LFB to do next hop action.
                  </synopsis>
                  <product>
                    <frameProduced>
                        <ref>IPv4Unicast</ref>
                    </frameProduced>
                    <metadataProduced>
                        <ref>HopSelector</ref>
                    </metadataProduced>
                  </product>
              </outputPort>
              <outputPort group="false">
                  <name>ECMPOut</name>
                  <synopsis>
```

```
                  The port to output packets needing further ECMP
                  processing. A downstream ECMP processing LFB is usually
                  followed to the port. If ECMP is not required, no
                  downstream LFB may be connected to the port.
                </synopsis>
                <product>
                  <frameProduced>
                     <ref>IPv4Unicast</ref>
                  </frameProduced>
                  <metadataProduced>
                     <ref>HopSelector</ref>
                  </metadataProduced>
                </product>
              </outputPort>
              <outputPort group="false">
                <name>ExceptionOut</name>
                <synopsis>
                  The port to output all packets with exceptional cases
                  happened during LPM process. An ExceptionID metadata
                  is associated to indicate what caused the exception.
                </synopsis>
                <product>
                  <frameProduced>
                     <ref>IPv4Unicast</ref>
                  </frameProduced>
                  <metadataProduced>
                     <ref>ExceptionID</ref>
                  </metadataProduced>
                </product>
              </outputPort>
            </outputPorts>
            <components>
              <component componentID="1" access="read-write">
                <name>IPv4PrefixTable</name>
                <synopsis>
                  A table for IPv4 Longest Prefix Match(LPM). 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>
                <typeRef>IPv4PrefixTableType</typeRef>
              </component>
              <component componentID="2" access="read-reset">
                <name>IPv4UcastLPMStats</name>
                <synopsis>
                  The statistics information for the IPv4 unicast LPM
                  process in the LFB.
                </synopsis>
```

```
                <optional/>
                <typeRef>IPv4UcastLPMStatsType</typeRef>
             </component>
          </components>
      </LFBClassDef>
      <LFBClassDef LFBClassID="11">
          <name>IPv6UcastLPM</name>
          <synopsis>
            The IPv6UcastLPM LFB abstracts the IPv6 unicast Longest
            Prefix Match (LPM) process. This LFB supports to implement
            equal-cost multi-path routing (ECMP) and reverse path
            forwarding (RPF).
          </synopsis>
          <version>1.0</version>
          <inputPorts>
             <inputPort group="false">
                <name>PktsIn</name>
                <synopsis>
                  A port for input of packets to be processed. IPv6
                  unicast packets are expected.
                </synopsis>
                <expectation>
                <frameExpected>
                   <ref>IPv6Unicast</ref>
                </frameExpected>
                </expectation>
             </inputPort>
          </inputPorts>
          <outputPorts>
             <outputPort group="false">
                <name>NormalOut</name>
                <synopsis>
                  An output port to output IPv6 unicast packets
                  successfully passed the LPM lookup. A HopSelector
                  metadata is produced to associate every output packet
                  for downstream LFB to do next hop action.
                </synopsis>
                <product>
                   <frameProduced>
                      <ref>IPv6Unicast</ref>
                   </frameProduced>
                   <metadataProduced>
                      <ref>HopSelector</ref>
                   </metadataProduced>
                </product>
             </outputPort>
             <outputPort group="false">
                <name>ECMPOut</name>
```

```
        <synopsis>
          The port to output packets needing further ECMP
          processing. A downstream ECMP processing LFB is usually
          followed to the port. If ECMP is not required, no
          downstream LFB may be connected to the port.
        </synopsis>
        <product>
          <frameProduced>
            <ref>IPv6Unicast</ref>
          </frameProduced>
          <metadataProduced>
            <ref>HopSelector</ref>
          </metadataProduced>
        </product>
    </outputPort>
    <outputPort group="false">
        <name>ExceptionOut</name>
        <synopsis>
          The port to output all packets with exceptional cases
          happened during LPM process. An ExceptionID metadata
          is associated to indicate what caused the exception.
        </synopsis>
        <product>
          <frameProduced>
            <ref>IPv6Unicast</ref>
          </frameProduced>
          <metadataProduced>
            <ref>ExceptionID</ref>
          </metadataProduced>
        </product>
    </outputPort>
  </outputPorts>
  <components>
    <component componentID="1" access="read-write">
      <name>IPv6PrefixTable</name>
      <synopsis>
        A table for IPv6 Longest Prefix Match(LPM). The
        destination IPv6 address of every input packet is
        used as a search key to look up the table to find
        out a next hop selector.
      </synopsis>
      <typeRef>IPv6PrefixTableType</typeRef>
    </component>
    <component componentID="2" access="read-reset">
      <name>IPv6UcastLPMStats</name>
      <synopsis>
        The statistics information for the IPv6 unicast LPM
        process in the LFB.
```

```
              </synopsis>
              <optional/>
              <typeRef>IPv6UcastLPMStatsType</typeRef>
            </component>
        </components>
    </LFBClassDef>
    <LFBClassDef LFBClassID="12">
        <name>IPv4NextHop</name>
        <synopsis>
          The IPv4NextHop LFB abstracts the process of next hop
          information application to IPv4 packets. It receives an IPv4
          packet with an associated next hop identifier (HopSelector),
          uses the identifier as a table index to look up a next hop
          table to find an appropriate output port. The data
          processing also involves the forwarding TTL decrement and
          IP checksum recalculation.
        </synopsis>
        <version>1.0</version>
        <inputPorts>
           <inputPort group="false">
              <name>PktsIn</name>
              <synopsis>
                A port for input of unicast IPv4 packets, along with
                a HopSelector metadata.
              </synopsis>
              <expectation>
              <frameExpected>
                  <ref>IPv4Unicast</ref>
              </frameExpected>
              <metadataExpected>
                  <ref>HopSelector</ref>
              </metadataExpected>
              </expectation>
           </inputPort>
        </inputPorts>
        <outputPorts>
           <outputPort group="true">
              <name>SuccessOut</name>
              <synopsis>
                The group port for output of packets who successfully
                found next hop information. Some metadata are
                associated with every packet.
              </synopsis>
              <product>
                 <frameProduced>
                    <ref>IPv4Unicast</ref>
                 </frameProduced>
                 <metadataProduced>
```

```
                      <ref>L3PortID</ref>
                      <ref>NextHopIPv4Addr</ref>
                      <ref availability="conditional">
                      MediaEncapInfoIndex</ref>
                  </metadataProduced>
               </product>
           </outputPort>
           <outputPort group="false">
              <name>ExceptionOut</name>
              <synopsis>
                The output port for packets with exceptional or
                failure case. An ExceptionID metadata indicates what
                caused the case.
              </synopsis>
              <product>
                 <frameProduced>
                     <ref>IPv4Unicast</ref>
                 </frameProduced>
                 <metadataProduced>
                     <ref>ExceptionID</ref>
                 </metadataProduced>
              </product>
           </outputPort>
        </outputPorts>
        <components>
           <component componentID="1">
              <name>IPv4NextHopTable</name>
              <synopsis>
                The IPv4NextHopTable component. A
                HopSelector is used to match the table index
                to find out a row which contains the next hop
                information result.
              </synopsis>
              <typeRef>IPv4NextHopTableType</typeRef>
           </component>
        </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="13">
        <name>IPv6NextHop</name>
        <synopsis>
          The LFB abstracts the process of next hop information
          application to IPv6 packets. It receives an IPv6 packet
          with an associated next hop identifier (HopSelector),
          uses the identifier as a table index to look up a next hop
          table to find an appropriate output port.
        </synopsis>
        <version>1.0</version>
        <inputPorts>
```

```
            <inputPort group="false">
               <name>PktsIn</name>
               <synopsis>
                 A port for input of unicast IPv6 packets, along with
                 a HopSelector metadata.
                </synopsis>
               <expectation>
               <frameExpected>
                   <ref>IPv6Unicast</ref>
               </frameExpected>
               <metadataExpected>
                   <ref>HopSelector</ref>
               </metadataExpected>
               </expectation>
            </inputPort>
         </inputPorts>
         <outputPorts>
            <outputPort group="true">
               <name>SuccessOut</name>
               <synopsis>
                 The group port for output of packets who successfully
                 found next hop information. Some metadata are
                 associated with every packet.
                </synopsis>
               <product>
                  <frameProduced>
                     <ref>IPv6Unicast</ref>
                  </frameProduced>
                  <metadataProduced>
                     <ref>L3PortID</ref>
                     <ref>NextHopIPv6Addr</ref>
                     <ref availability="conditional">
                     MediaEncapInfoIndex</ref>
                  </metadataProduced>
               </product>
            </outputPort>
            <outputPort group="false">
               <name>ExceptionOut</name>
               <synopsis>
                 The output port for packets with exceptional or
                 failure case. An ExceptionID metadata indicates what
                 caused the case.
               </synopsis>
               <product>
                  <frameProduced>
                     <ref>IPv6Unicast</ref>
                  </frameProduced>
                  <metadataProduced>
```

```
                        <ref>ExceptionID</ref>
                      </metadataProduced>
                  </product>
              </outputPort>
          </outputPorts>
          <components>
              <component componentID="1">
                  <name>IPv6NextHopTable</name>
                  <synopsis>
                    The IPv6NextHopTable component. A HopSelector is used
                    to match the table index to find out a row which
                    contains the next hop information result.
                  </synopsis>
                  <typeRef>IPv6NextHopTableType</typeRef>
              </component>
          </components>
      </LFBClassDef>
      <LFBClassDef LFBClassID="14">
          <name>RedirectIn</name>
          <synopsis>
            The RedirectIn LFB abstracts the process for the ForCES CE to
            inject data packets into the ForCES FE LFBs.
          </synopsis>
          <version>1.0</version>
          <outputPorts>
              <outputPort group="true">
                  <name>PktsOut</name>
                  <synopsis>
                    The output port of RedirectIn LFB, which is defined as
                    a group port type. From LFB topology point of view,
                    the RedirectIn LFB acts as a source point for data
                    packets coming from CE, therefore the LFB is
                    defined with a singleton output port (and no input
                    port).
                  </synopsis>
                  <product>
                      <frameProduced>
                          <ref>Arbitrary</ref>
                      </frameProduced>
                  </product>
              </outputPort>
          </outputPorts>
          <components>
              <component componentID="1">
                  <name>NumPacketsReceived</name>
                  <synopsis>
                    Number of packets received from CE.
                  </synopsis>
```

```
              <optional/>
              <typeRef>uint64</typeRef>
           </component>
        </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="15">
        <name>RedirectOut</name>
        <synopsis>
          The RedirectOut LFB abstracts the process for LFBs in a
          ForCES FE to deliver data packets to the ForCES CE.
        </synopsis>
        <version>1.0</version>
        <inputPorts>
           <inputPort group="false">
              <name>PktsIn</name>
              <synopsis>
                The input port for the RedirectOut LFB. From the LFB's
                topology point of view, the RedirectOut LFB acts as a
                sink point for data packets going to the CE, therefore
                RedirectOut LFB is defined with a singleton input
                port (and no output port).
              </synopsis>
              <expectation>
                 <frameExpected>
                    <ref>Arbitrary</ref>
                 </frameExpected>
              </expectation>
           </inputPort>
        </inputPorts>
        <components>
           <component componentID="1">
              <name>NumPacketsSent</name>
              <synopsis>
                Numble of packets sent to CE.
              </synopsis>
              <optional/>
              <typeRef>uint64</typeRef>
           </component>
        </components>
     </LFBClassDef>
     <LFBClassDef LFBClassID="16">
        <name>BasicMetadataDispatch</name>
        <synopsis>
          The BasicMetadataDispatch LFB is defined to abstract the
          process by which packets are dispatched to various output
          paths based on associated metadata value. Current version of
          the LFB only allows the metadata value be 32-bits integer.
        </synopsis>
```

```
<version>1.0</version>
<inputPorts>
   <inputPort>
      <name>PktsIn</name>
      <synopsis>
        The packet input port for dispatching. Every input
        packet should be associated with a metadata that will
        be used by the LFB to do the dispatch.
      </synopsis>
      <expectation>
         <frameExpected>
            <ref>Arbitrary</ref>
         </frameExpected>
         <metadataExpected>
            <ref>Arbitrary</ref>
         </metadataExpected>
      </expectation>
   </inputPort>
</inputPorts>
<outputPorts>
   <outputPort group="true">
      <name>PktsOut</name>
      <synopsis>
        The group output port which outputs dispatching
        results. A packet with its associated metadata having
        found an OutputIndex by successfully looking up the
        dispatch table will be output to the group port
        instance with the corresponding index.
      </synopsis>
      <product>
         <frameProduced>
            <ref>Arbitrary</ref>
         </frameProduced>
      </product>
   </outputPort>
   <outputPort group="false">
      <name>ExceptionOut</name>
      <synopsis>
        The output port which outputs packets which failed
        to process. An ExceptionID metadata indicates what
        caused the exception.
      </synopsis>
      <product>
         <frameProduced>
            <ref>Arbitrary</ref>
         </frameProduced>
         <metadataProduced>
            <ref>ExceptionID</ref>
```

```
                </metadataProduced>
              </product>
          </outputPort>
       </outputPorts>
       <components>
          <component access="read-write" componentID="1">
             <name>MetadataID</name>
             <synopsis>
               The ID of the metadata to be
               used for dispatching packets.
             </synopsis>
             <typeRef>uint32</typeRef>
          </component>
          <component access="read-write" componentID="2">
             <name>MetadataDispatchTable</name>
             <synopsis>
               The MetadataDispatchTable component, which contains
               entries of a metadata value and an output index,
               specifying that a packet with the metadata value must
               go out from the instance with the output index of the
               LFB group output port.
             </synopsis>
             <typeRef>MetadataDispatchTableType</typeRef>
          </component>
       </components>
    </LFBClassDef>
    <LFBClassDef LFBClassID="17">
       <name>GenericScheduler</name>
       <synopsis>
         This is a preliminary generic scheduler LFB abstracting
         a simple scheduling process, which may be used as a
         basic LFB to construct a more complex scheduler LFB.
       </synopsis>
       <version>1.0</version>
       <inputPorts>
          <inputPort group="true">
             <name>PktsIn</name>
             <synopsis>
               The group input port of the LFB. Inside the LFB,
               each instance of the group port is connected to
               a queue marked with a queue ID, whose value is
               index of the port instance.
             </synopsis>
             <expectation>
                <frameExpected>
                   <ref>Arbitrary</ref>
                </frameExpected>
             </expectation>
```

```
            </inputPort>
         </inputPorts>
         <outputPorts>
            <outputPort>
               <name>PktsOut</name>
               <synopsis>
                 The output port of the LFB. Scheduled packets are
                 output from the port.
               </synopsis>
               <product>
                  <frameProduced>
                     <ref>Arbitrary</ref>
                  </frameProduced>
               </product>
            </outputPort>
         </outputPorts>
         <components>
            <component access="read-write" componentID="1">
               <name>SchedulingDiscipline</name>
               <synopsis>
                 The SchedulingDiscipline component, which is for the
                 CE to specify a scheduling discipline to the LFB.
               </synopsis>
               <typeRef>SchdDisciplineType</typeRef>
               <defaultValue>1</defaultValue>
            </component>
            <component access="read-only" componentID="2">
               <name>QueueStats</name>
               <synopsis>
                 The QueueStats component, which is defined to allow
                 CE to query every queue statistics in the scheduler.
               </synopsis>
               <optional/>
               <typeRef>QueueStatsTableType</typeRef>
            </component>
         </components>
         <capabilities>
            <capability componentID="30">
               <name>QueueLenLimit</name>
               <synopsis>
                 The QueueLenLimit capability, which specifies
                 maximum length of each queue. The length unit is in
                 bytes.
               </synopsis>
               <typeRef>uint32</typeRef>
            </capability>
         </capabilities>
      </LFBClassDef>
```

```
    </LFBClassDefs>
</LFBLibrary>
```

[7](#). **LFB Class Use Cases**

   This section demonstrates examples on how the LFB classes defined by
   the Base LFB library in [Section 6](#) can be applied to achieve some
   typical router functions. The functions demonstrated are:

   o IPv4 forwarding

   o ARP processing

   It is assumed the LFB topology on the FE described has already been
   established by the CE and maps to the use cases illustrated in this
   section.

   The use cases demonstrated in this section are mere examples and by
   no means should be treated as the only way one would construct router
   functionality from LFBs; based on the capability of the FE(s), a CE
   should be able to express different NE applications.

[7.1](#). **IPv4 Forwarding**

   Figure 2 shows the typical LFB processing path for an IPv4 unicast
   forwarding case with Ethernet media interfaces by use of the base LFB
   classes. Note that in the figure, to focus on the IP forwarding
   function, some inputs or outputs of LFBs that are not related to the
   IPv4 forwarding function are not shown. For example, an
   EtherClassifier LFB normally has two output ports: a "ClassifyOut"
   group output port and a "ExceptionOut" singleton output port, with
   the group port contains various port instances according to various
   classified packet types([Section 5.1.3](#)). While in this figure, only
   the IPv4 and IPv6 packet output port instances are shown for
   displaying the mere IPv4 forwarding processing function.

```
      +-----+                    +------+
      |     |                    |      |
      |     |<---------------|Ether |<----------------------------+
      |     |                |MACOut|                             |
      |     |                |      |                             |
      |Ether|                +------+                             |
      |PHY  |                                                     |
      |Cop  |            +---+                                    |
      |#1   |   +-----+  |   |----->IPv6 Packets                  |
      |     |   |     |  |   |  |                                 |
      |     |   |Ether|  |   |  | IPv4 Packets                    |
      |     |->|MACIn|-->|   |-+  +----+                          |
      +-----+   |     |  |   | |  |    |---> Multicast Packets    |
                +-----+  +---+ |  |    |          +-----+  +---+   |
                       Ether +->|  |------->|     |   |  |   |   |
          .          Classifier|  |   |Unicast |IPv4 |  |   |   |
          .                    |  |   |Packets |Ucast|->|   |--+ |
          .                    |  +----+        |LPM  |  |   | | |
                      +---+ |    IPv4            +-----+  +---+ | |
            +-----+   |   | |  |  Validator              IPv4 | |
            |     |   |   | |  |                        NextHop| |
      +-----+  |Ether|   |  |-+ IPv4 Packets                  | |
      |     |->|MACIn|-->|   |                                 | |
      |     |   |     |  |   |  |----->IPv6 Packets             | |
      |Ether|   +-----+  +---+                                 | |
      |PHY  |          Ether              +----+               | |
      |Cop  |          Classifier         |    |   +-------+    | |
      |#n   |            +------+          |    |   |Ether  |   | |
      |     |            |      |          |    |<--|Encap  |<-+ |
      |     |            |      |  |<------|    |   |       |    |
      |     |<---------------|Ether |    ...|   |   +-------+    |
      |     |            |MACOut|  +---|    |                    |
      |     |            |      |  |  | +----+                   |
      +-----+            +------+  | BasicMetadataDispatch       |
                                +----------->-------------+
```
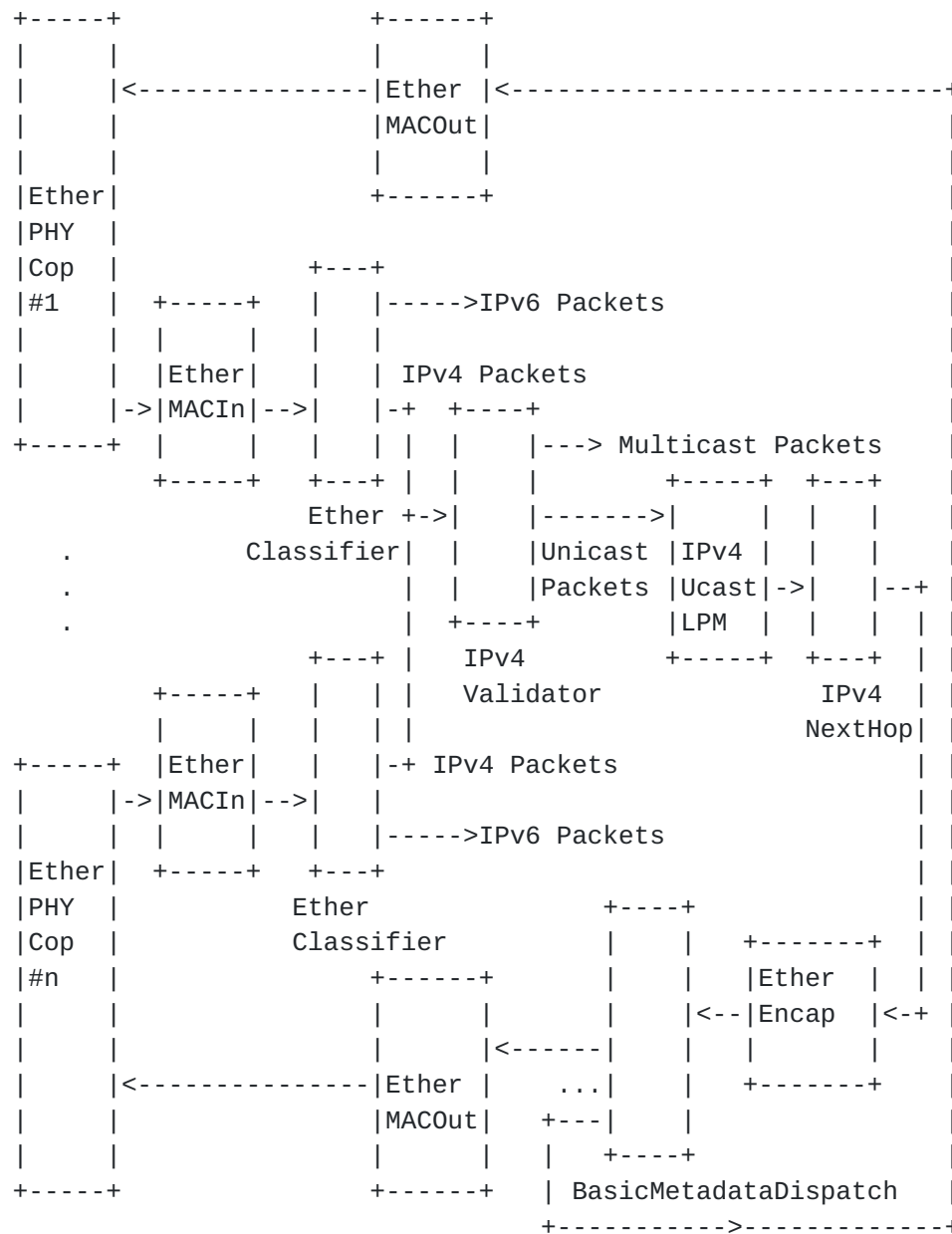
                  Figure 2: LFB use case for IPv4 forwarding

   In the LFB use case, a number of EtherPHYCop LFB([Section 5.1.1](#))
   instances are used to describe physical layer functions of the ports.
   PHYPortID metadata is generated by EtherPHYCop LFB and is used by all
   the subsequent downstream LFBs. An EtherMACIn LFB([Section 5.1.2](#)),
   which describe the MAC layer processing, follows every EtherPHYCop
   LFB. The EtherMACIn LFB may do a locality check of MAC addresses if
   the CE configures the appropriate EtherMACIn LFB component.

   Ethernet packets out of the EtherMACIn LFB are sent to an

EtherClassifier LFB (Section 5.1.3) to be decapsulated and classified
into network layer types like IPv4, IPv6, ARP, etc. In the example
use case, every physical Ethernet interface is associated with one
Classifier instance; although not illustrated, it is also feasible
that all physical interfaces are associated with only one Ethernet
Classifier instance.

EtherClassifier uses the PHYPortID metadata, the Ethernet type of the
input packet, and VlanID (if present in the input Ethernet packets),
to decide the packet network layer type and the LFB output port to
the downstream LFB. The EtherClassifier LFB also assigns a new
logical port ID metadata to the packet for later use. The
EtherClassifier may also generate some new metadata for every packet
like EtherType, SrcMAC, DstMAC, LogicPortID, etc for consumption by
downstream LFBs.

If a packet is classified as an IPv4 packet, it is sent downstream to
an IPv4Validator LFB (Section 5.2.1) to validate the IPv4 packet. In
the validator LFB, IPv4 packets are validated and are additionally
classified into either IPv4 unicast packets or multicast packets.
IPv4 unicast packets are sent to downstream to the IPv4UcastLPM LFB
(Section 5.3.1).

The IPv4UcastLPM LFB is where the longest prefix match decision is
made, and a next hop selection is selected. The next hop ID metadata
is generated by the IPv4UcastLPM LFB to be consumed downstream by the
IPv4NextHop LFB (Section 5.3.2).

The IPv4NextHop LFB uses the next hop ID metadata to do derive where
the packet is to go next and the media encapsulation type for the
port, etc. The IPv4NextHop LFB generates the L3PortID metadata used
to identify a next hop output physical/logical port. In the example
use case, the next hop output port is an Ethernet type; as a result,
the packet and its L3 port ID metadata are sent downstream to an
EtherEncap LFB (Section 5.1.4).

The EtherEncap LFB encapsulates the incoming packet into an Ethernet
frame. A BasicMetadataDispatch LFB (Section 5.5.1) follows the
EtherEncap LFB. The BasicMetadataDispatch LFB is where packets are
finally dispatched to different output physical/logical ports based
on the L3PortID metadata sent to the LFB.

7.2. ARP processing

Figure 3 shows the processing path for ARP protocol in the case the
CE implements the ARP processing function. By no means is this the
only way ARP processing could be achieved; as an example ARP
processing could happen at the FE - but that discussion is out of
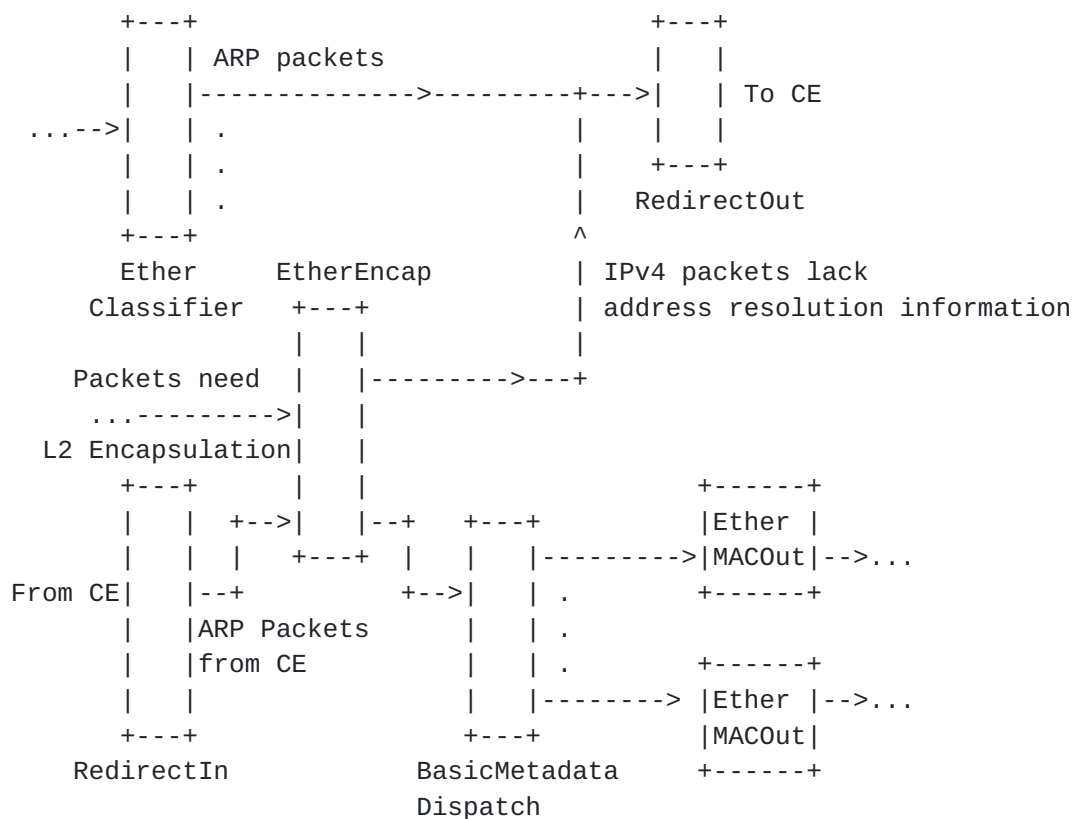
scope for this use case.

```
        +---+                              +---+
        |   | ARP packets                  |   |
        |   |-------------->---------+--->| | To CE
  ...-->|   | .                      |   |   |
        |   | .                      |   +---+
        |   | .                      | RedirectOut
        +---+                        ^
        Ether    EtherEncap          | IPv4 packets lack
       Classifier  +---+             | address resolution information
                   |   |             |
      Packets need |   |--------->---+
      ...--------->|   |
     L2 Encapsulation|   |
        +---+     |   |                    +------+
        |   | +-->|   |--+   +---+          |Ether |
        |   | |   +---+  |   |   |--------->|MACOut|-->...
   From CE|   |--+          +-->|   | .         +------+
        |   |ARP Packets    |   | .
        |   |from CE        |   | .         +------+
        |   |               |   |--------> |Ether |-->...
        +---+               +---+          |MACOut|
      RedirectIn          BasicMetadata    +------+
                          Dispatch
```

                    Figure 3: LFB use case for ARP

   There are two ways ARP processing could be triggered in the CE as
   illustrated in Figure 3:

   o ARP packets arriving from outside of the NE.

   o IPV4 packets failing to resolve within the FE.

   ARP packets from network interfaces are filtered out by
   EtherClassifier LFB. The classified ARP packets and associated
   metadata are then sent downstream to the RedirectOut LFB
   (Section 5.4.2) to be transported to CE.

   The EtherEncap LFB, as described in Section 5.1.4, receives packets
   that need Ethernet L2 encapsulating. When the EtherEncap LFB fails
   to find the necessary L2 Ethernet information to encapsulate the
   packet with, it outputs the packet to its ExceptionOut LFB port.
   Downstream to EtherEncap LFB's ExceptionOut LFB port is the
   RedirectOut LFB which transports the packet to the CE (Section 5.1.4
   on EtherEncap LFB for details).

To achieve its goal, the CE needs to generate ARP request and
response packets and send them to external (to the NE) networks. ARP
request and response packets from the CE are redirected to an FE via
a RedirectIn LFB (Section 5.4.1).

As was the case with forwarded IPv4 packets, outgoing ARP packets are
also encapsulated to Ethernet format by the EtherEncap LFB, and then
dispatched to different interfaces via a BasicMetadataDispatch LFB.
The BasicMetadataDispatch LFB dispatches the packets according to the
L3PortID metadata included in every ARP packet sent from CE.

8. Contributors

   The authors would like to thank Jamal Hadi Salim, Ligang Dong, and
   Fenggen Jia who made major contributions to the development of this
   document.

      Jamal Hadi Salim
      Mojatatu Networks
      Ottawa, Ontario
      Canada
      Email: hadi@mojatatu.com

      Ligang Dong
      Zhejiang Gongshang University
      149 Jiaogong Road
      Hangzhou 310035
      P.R.China
      Phone: +86-571-28877751
      EMail: donglg@mail.zjgsu.edu.cn

      Fenggen Jia
      National Digital Switching Center(NDSC)
      Jianxue Road
      Zhengzhou 452000
      P.R.China
      EMail: jfg@mail.ndsc.com.cn

9. Acknowledgements

This document is based on earlier documents from Joel Halpern, Ligang Dong, Fenggen Jia and Weiming Wang.

10. IANA Considerations

   IANA has created a registry of ForCES LFB Class Names and the
   corresponding ForCES LFB Class Identifiers, with the location of the
   definition of the ForCES LFB Class, in accordance with the rules to
   use the namespace.

   The LFB library in this document needs for unique class names and
   numeric class identifiers of all LFBs. Besides, this document also
   needs to define the following namespaces:

   o Metadata ID, defined in Section 4.3 and Section 4.4

   o Exception ID, defined in Section 4.4

   o Validate Error ID, defined in Section 4.4

10.1. LFB Class Names and LFB Class Identifiers

   LFB classes defined by this document belongs to IETF defined LFBs by
   Standard Track RFCs. According to IANA, the identifier namespace for
   these LFB classes is from 3 to 65535.

   The assignment of LFB class names and LFB class identifiers is as in
   the following table.

LFB Class Identifier	LFB Class Name	Description	Reference
3	EtherPHYCop	Define an Ethernet port abstracted at physical layer.	RFC????(this document) Section 5.1.1
4	EtherMACIn	Define an Ethernet input port at MAC data link layer.	RFC???? Section 5.1.2
5	EtherClassifier	Define the process to decapsulate Ethernet packets and classify the packets.	RFC???? Section 5.1.3
6	EtherEncap	Define the process to encapsulate IP packets to Ethernet packets.	RFC???? Section 5.1.4

7	EtherMACOut	Define an Ethernet output port at MAC data link layer.	RFC ???? Section 5.1.5
8	IPv4Validator	Perform IPv4 packets validation.	RFC ???? Section 5.2.1
9	IPv6Validator	Perform IPv6 packets validation.	RFC ???? Section 5.2.2
10	IPv4UcastLPM	Perform IPv4 Longest Prefix Match Lookup.	RFC ???? Section 5.3.1
11	IPv6UcastLPM	Perform IPv6 Longest Prefix Match Lookup.	RFC ???? Section 5.3.3
12	IPv4NextHop	Define the process of selecting Ipv4 next hop action.	RFC ??? Section 5.3.2
13	IPv6NextHop	Define the process of selecting Ipv6 next hop action.	RFC ??? Section 5.3.4
14	RedirectIn	Define the process for CE to inject data packets into FE LFB topology.	RFC ??? Section 5.4.1
15	RedirectOut	Define the process for LFBs in FE to deliver data packets to CE.	RFC ??? Section 5.4.2
16	BasicMetadata Dispatch	Dispatch input packets to a group output according to a metadata	RFC ??? Section 5.5.1
17	Generic Scheduler	Define a preliminary generic scheduling process.	RFC ???? Section 5.5.2

                               Table 1

10.2. Metadata ID

   The Metadata ID namespace is 32 bits long. The following is the
   guideline for managing the namespace.

   Metadata ID 0x00000000-0x7FFFFFFF
      Metadata with IDs in this range are Specification Required
      [RFC5226]. A metadata ID using this range MUST be documented in
      an RFC or other permanent and readily available references.

      Values assigned by this specification:

   +--------------+------------------------+-------------------------+
   | Value | Name | Definition |
   +--------------+------------------------+-------------------------+
0x00000001	PHYPortID	See Section 4.4
0x00000002	SrcMAC	See Section 4.4
0x00000003	DstMAC	See Section 4.4
0x00000004	LogicalPortID	See Section 4.4
0x00000005	EtherType	See Section 4.4
0x00000006	VlanID	See Section 4.4
0x00000007	VlanPriority	See Section 4.4
0x00000008	NextHopIPv4Addr	See Section 4.4
0x00000009	NextHopIPv6Addr	See Section 4.4
0x0000000A	HopSelector	See Section 4.4
0x0000000B	ExceptionID	See Section 4.4
0x0000000C	ValidateErrorID	See Section 4.4
0x0000000D	L3PortID	See Section 4.4
0x0000000E	RedirectIndex	See Section 4.4
0x0000000F	MediaEncapInfoIndex	See Section 4.4
   +--------------+------------------------+-------------------------+

                                 Table 2

   Metadata ID 0x80000000-0xFFFFFFFF
      Metadata IDs in this range are reserved for vendor private
      extensions and are the responsibility of individuals.

10.3. Exception ID

   The Exception ID namespace is 32 bits long. The following is the
   guideline for managing the namespace.

Exception ID 0x00000000-0x7FFFFFFF
   Exception IDs in this range are Specification Required [RFC5226].
   An exception ID using this range MUST be documented in an RFC or
   other permanent and readily available references.

   Values assigned by this specification:


```
      +--------------+--------------------------------+-----------------+
      |    Value     |             Name               |   Definition    |
      +--------------+--------------------------------+-----------------+
      |  0x00000000  |   AnyUnrecognizedExceptionCase  | See Section 4.4 |
      |  0x00000001  |        ClassifyNoMatching       | See Section 4.4 |
      |  0x00000002  |    MediaEncapInfoIndexInvalid   | See Section 4.4 |
      |  0x00000003  |       EncapTableLookupFailed    | See Section 4.4 |
      |  0x00000004  |              BadTTL             | See Section 4.4 |
      |  0x00000005  |     IPv4HeaderLengthMismatch    | See Section 4.4 |
      |  0x00000006  |         RouterAlertOptions      | See Section 4.4 |
      |  0x00000007  |          IPv6HopLimitZero       | See Section 4.4 |
      |  0x00000008  |         IPv6NextHeaderHBH       | See Section 4.4 |
      |  0x00000009  |       SrcAddressExecption       | See Section 4.4 |
      |  0x0000000A  |       DstAddressExecption       | See Section 4.4 |
      |  0x0000000B  |         LPMLookupFailed         | See Section 4.4 |
      |  0x0000000C  |        HopSelectorInvalid       | See Section 4.4 |
      |  0x0000000D  |        NextHopLookupFailed      | See Section 4.4 |
      |  0x0000000E  |           FragRequired          | See Section 4.4 |
      |  0x0000000F  |        MetadataNoMatching       | See Section 4.4 |
      +--------------+--------------------------------+-----------------+
```

                              Table 3

Exception ID 0x80000000-0xFFFFFFFF
   Exception IDs in this range are reserved for vendor private
   extensions and are the responsibility of individuals.

10.4. Validate Error ID

   The Validate Error ID namespace is 32 bits long. The following is
   the guideline for managing the namespace.

   Validate Error ID 0x00000000-0x7FFFFFFF
      Validate Error IDs in this range are Specification Required
      [RFC5226]. A Validate Error ID using this range MUST be
      documented in an RFC or other permanent and readily available
      references.

Values assigned by this specification:

```
+--------------+-------------------------------+------------------+
|   Value      |             Name              |   Definition     |
+--------------+-------------------------------+------------------+
|  0x00000000  | AnyUnrecognizedValidateErrorCase| See Section 4.4 |
|  0x00000001  |        InvalidIPv4PacketSize  | See Section 4.4  |
|  0x00000002  |          NotIPv4Packet        | See Section 4.4  |
|  0x00000003  |     InvalidIPv4HeaderLengthSize | See Section 4.4 |
|  0x00000004  |     InvalidIPv4LengthFieldSize  | See Section 4.4 |
|  0x00000005  |          InvalidIPv4Checksum  | See Section 4.4  |
|  0x00000006  |       InvalidIPv4SrcAddr       | See Section 4.4  |
|  0x00000007  |       InvalidIPv4DstAddr       | See Section 4.4  |
|  0x00000008  |       InvalidIPv6PakcetSize    | See Section 4.4  |
|  0x00000009  |          NotIPv6Packet        | See Section 4.4  |
|  0x0000000A  |       InvalidIPv6SrcAddr       | See Section 4.4  |
|  0x0000000B  |       InvalidIPv6DstAddr       | See Section 4.4  |
+--------------+-------------------------------+------------------+
```
                            Table 4

Validate Error ID 0x80000000-0xFFFFFFFF
    Validate Error IDs in this range are reserved for vendor private
    extensions and are the responsibility of individuals.

11. Security Considerations

   The ForCES framework document [RFC3746] provides a description of the
   security needs for the overall ForCES architecture. For example, the
   ForCES protocol entities must be authenticated per the ForCES
   requirements before they can access the information elements
   described in this document via ForCES. The ForCES protocol document
   [RFC5810] includes a comprehensive set of security mechanisms and
   which implementations are required to support, and which deployments
   can use to meet these needs. The LFBs defined in this document are
   similar to other LFBs modeled by the FE model [RFC5812]. In
   particular, they have the same security properties. Thus the
   security mechanisms and considerations from the ForCES protocol
   document [RFC5810] apply and address the issues.

12. References

12.1. Normative References

[RFC5810] Doria, A., Hadi Salim, J., Haas, R., Khosravi, H., Wang,
           W., Dong, L., Gopal, R., and J. Halpern, "Forwarding and
           Control Element Separation (ForCES) Protocol
           Specification", RFC 5810, March 2010.

[RFC5812] Halpern, J. and J. Hadi Salim, "Forwarding and Control
           Element Separation (ForCES) Forwarding Element Model",
           RFC 5812, March 2010.

12.2. Informative References

[RFC1122] Braden, R., "Requirements for Internet Hosts -
           Communication Layers", STD 3, RFC 1122, October 1989.

[RFC1812] Baker, F., "Requirements for IP Version 4 Routers",
           RFC 1812, June 1995.

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

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6
           (IPv6) Specification", RFC 2460, December 1998.

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J.
           Schoenwaelder, Ed., "Structure of Management Information
           Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629,
           June 1999.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC
           Text on Security Considerations", BCP 72, RFC 3552,
           July 2003.

[RFC3654] Khosravi, H. and T. Anderson, "Requirements for Separation
           of IP Control and Forwarding", RFC 3654, November 2003.

[RFC3746] Yang, L., Dantu, R., Anderson, T., and R. Gopal,
           "Forwarding and Control Element Separation (ForCES)
           Framework", RFC 3746, April 2004.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
           IANA Considerations Section in RFCs", BCP 26, RFC 5226,
           May 2008.

Authors' Addresses

   Weiming Wang
   Zhejiang Gongshang University
   18 Xuezheng Str., Xiasha University Town
   Hangzhou,   310018
   P.R.China

   Phone: +86 571 28877721
   Email: wmwang@zjsu.edu.cn

   Evangelos Haleplidis
   University of Patras
   Patras,
   Greece

   Email: ehalep@ece.upatras.gr

   Kentaro Ogawa
   NTT Corporation
   Tokyo,
   Japan

   Email: ogawa.kentaro@lab.ntt.co.jp

   Chuanhuang Li
   Hangzhou DPtech
   6th Floor, Zhongcai Group, 68 Tonghe Road, Binjiang District
   Hangzhou,   310051
   P.R.China

   Email: chuanhuang_li@zjsu.edu.cn

   Halpern Joel
   Ericsson
   P.O. Box 6049
   Leesburg,   20178
   VA

   Phone: +1 703 371 3043
   Email: joel.halpern@ericsson.com