Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: August 9, 2014

E. Haleplidis
University of Patras
J. Halpern
Ericsson

February 5, 2014

ForCES Packet Parallelization
draft-ietf-forces-packet-parallelization-00

Abstract

Forwarding and Control Element Separation (ForCES) defines an architectural framework and associated protocols to standardize information exchange between the control plane and the forwarding plane in a ForCES Network Element (ForCES NE). RFC5812 has defined the ForCES Model provides a formal way to represent the capabilities, state, and configuration of forwarding elements within the context of the ForCES protocol, so that control elements (CEs) can control the FEs accordingly. More specifically, the model describes the logical functions that are present in an FE, what capabilities these functions support, and how these functions are or can be interconnected.

Many network devices support parallel packet processing. This document describes how ForCES can model a network device's parallelization datapath.

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].

1.2. Definitions

This document follows the terminology defined by the ForCES Model in [RFC5812]. The required definitions are repeated below for clarity.

FE Model - The FE model is designed to model the logical processing functions of an FE. 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].

LFB (Logical Functional Block) Class (or type) - A template that represents a fine-grained, logically separable aspect of FE processing. Most LFBs relate to packet processing in the data path. LFB classes are the basic building blocks of the FE model.

LFB Instance - As a packet flows through an FE along a data path, it flows through one or multiple LFB instances, where each LFB is an instance of a specific LFB class. Multiple instances of the same LFB class can be present in an FE's data path. Note that we often refer to LFBs without distinguishing between an LFB class and LFB instance when we believe the implied reference is obvious for the given context.

LFB Model - The LFB model describes the content and structures in an LFB, plus the associated data definition. XML is used to provide a formal definition of the necessary structures for the modeling. Four types of information are defined in the LFB model. The core part of the LFB model is the LFB class definitions; the other three types of information define constructs associated with and used by the class definition. These are reusable data types, supported frame (packet) formats, and metadata.

Element - Element is generally used in this document in accordance with the XML usage of the term. It refers to an XML tagged part of an XML document. For a precise definition, please see the full set of XML specifications from the W3C. This term is included in this list for completeness because the ForCES formal model uses XML.

Attribute - Attribute is used in the ForCES formal modeling in accordance with standard XML usage of the term, i.e., to provide attribute information included in an XML tag.

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, but not how the per-packet state is implemented within actual hardware. Metadata is sent between the FE and the CE on redirect packets.

ForCES Component - A ForCES Component is a well-defined, uniquely identifiable and addressable ForCES model building block. A component has a 32-bit ID, name, type, and an optional synopsis description. These are often referred to simply as components.
LFB Component - An LFB component is a ForCES component that defines the Operational parameters of the LFBs that must be visible to the CEs.

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.

2. Introduction

A lot of network devices can process packets in a parallel manner. The ForCES Model [RFC5812] presents a formal way to describe the Forwarding Plane's datapath with Logical Function Blocks (LFBs) using XML. This document describes how packet parallelization can be described with the ForCES model.

The modelling concept has been influenced by Cilc [Cilc]. Cilc is a programming language that has been developed since 1994 at the MIT Laboratory to allow programmers to identify elements that can be executed in parallel. The two Cilc concepts used in this document is spawn and sync. Spawn being the place where parallel work can start and sync being the place where the parallel work finishes and must collect all parallel output.

3. Packet Parallelization

This document addresses the following two types of packet parallelization:

1. Flood - where a copy of a packet is sent to multiple LFBs to be processed in parallel.

2. Split - where the packet will be split in equal size chunks specified by the CE and sent to multiple LFB instances probably of the same LFB class to be processed in parallel.

It must be noted that the process of copying the packet in the Flood parallel type is implementation depended and is loosely defined here. An implementor may either decide to physical copy the packet and send all packets on the parallel paths, or may decide to logically copy the packet by simply sending for example pointers of the same packet provided that the necessary interlocks are taken into account. The implementor has to take into account the device's characteristics to decide which approach fits bets to the hardware.

Additionally in the split parallel type, while harder, the implementor may also decide to logically split the packet and send for example pointers to parts of the packet, provided that the necessary interlocks are managed.

This document introduces two LFBs that are used in before and after the parallelization occurs:

1. Splitter - similar to Cilc's spawn. An LFB that will split the path of a packet and be sent to multiple LFBs to be processed in parallel.

2. Merger - similar to Cilc's sync. An LFB that will receive packets or chunks of the same initial packet and merge them into one.

Both parallel packet distribution types can currently be achieved with the ForCES model. The splitter LFB has one group output that produces either chunks or packets to be sent to LFBs for processing and the merger LFB has one group input that expects either packets or chunks to aggregate all the parallel packets or chunks and produce a single packet. Figure 1 shows an simple example of a split parallel datapath along with the splitter and merger LFB. Figure 2 shows an example of a flood parallel datapath along with the splitter and merger LFB.

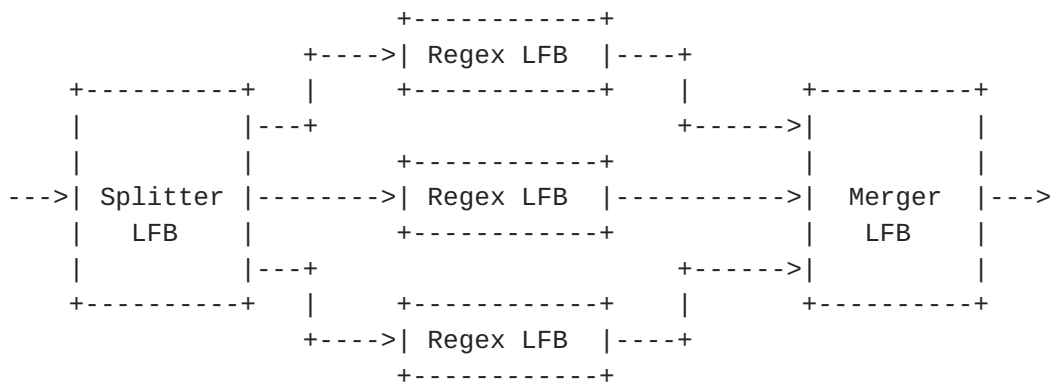

Figure 1: Simple split parallel processing

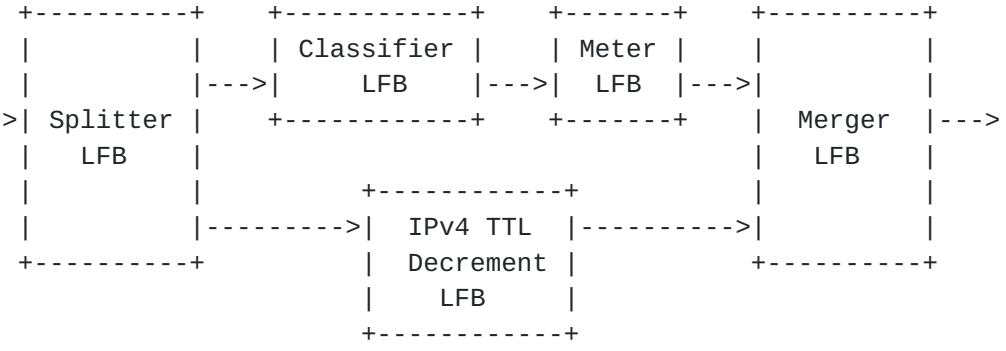

Figure 2: Simple flood parallel processing

This version of the modelling framework does not allow for nested parallel datapath topologies. This decision was reached by the authors and the ForCES working group as there was no strong use case or need at the time. This led to a more simple metadata definition needed to be transported between the splitter and the corresponding merger. If there is a need for nested parallel datapaths a new version of a splitter and merger will be needed to be defined as well as an augmentation to the defined metadata.

One important element to a developer is the ability to define which LFBs can be used in a parallel mode, with which other LFBs can they be parallelized with and the order of the LFBs can be assembled. This information must be accessible in the core LFBs. However instead of appending one more capability in the FEObject LFB and changing the FEObject LFB, we opted for an alternative. We introduced an additional core LFB, the CoreParallelization, that will not have input and output ports, but simply the capability necessary for LFB parallelization. If this LFB is not supported in the FEObjects LFB's SupportedLFBs component and not instantiated in the FE, that means that the FE does not support LFB parallelization.

The topology of the parallel datapath can be deferred and manipulated from the FEObject LFB's LFBTopology.

The CoreParallelization requires only one capability in order to specify each LFB that can be used in a parallel mode:

- The Name of the LFB.
- The Class ID of the LFB.
- The Version of the LFB.
- The number of instances that class can support in parallel.
- A list of LFB classes that can follow this LFB class in a pipeline for a parallel path.
- A list of LFB classes that can exist before this LFB class in a pipeline for a parallel path.
- A list of LFB classes that can process packets or chunks in parallel with this LFB class.

```
 <!-- Datatype -->
 <dataTypeDef>
    <name>ParallelLFBType</name>
    <synopsis>Table entry for parallel LFBs</synopsis>
    <struct>
       <component componentID="1">
          <name>LFBName</name>
          <synopsis>The name of an LFB Class</synopsis>
          <typeRef>string</typeRef>
       </component>
       <component componentID="2">
          <name>LFBClassID</name>
          <synopsis>The id of the LFB Class</synopsis>
          <typeRef>uint32</typeRef>
       </component>
       <component componentID="3">
          <name>LFBVersion</name>
          <synopsis>The version of the LFB Class used by this FE
          </synopsis>
          <typeRef>string</typeRef>
       </component>
       <component componentID="4">
          <name>LFBParallelOccurenceLimit</name>
          <synopsis>The upper limit of instances of the same
             parallel LFBs of this class</synopsis>
```

```
               <optional />
               <typeRef>uint32</typeRef>
            </component>
            <component componentID="5">
               <name>AllowedParallelAfters</name>
               <synopsis>List of LFB Classes that can follow this LFB
                  in a parallel pipeline</synopsis>
               <optional />
               <array>
                  <typeRef>uint32</typeRef>
               </array>
            </component>
            <component componentID="6">
               <name>AllowedParallelBefores</name>
               <synopsis>List of LFB Classes that this LFB class can
                  follow in a parallel pipeline</synopsis>
               <optional />
               <array>
                  <typeRef>uint32</typeRef>
               </array>
            </component>
            <component componentID="7">
               <name>AllowedParallel</name>
               <synopsis>List of LFB Classes that this LFB class be run
                  in parallel with</synopsis>
               <array>
                  <typeRef>uint32</typeRef>
               </array>
            </component>
         </struct>
      </dataTypeDef>

      <!-- Capability -->
            <capability componentID="32">
               <name>ParallelLFBs</name>
               <synopsis>List of all supported parallel LFBs</synopsis>
               <array type="Variable-size">
                  <typeRef>ParallelLFBType</typeRef>
               </array>
            </capability>
```

Figure 3: XML Definitions for CoreParallelization LFB

While the ForCES model cannot describe how the splitting or the merging is actually done as that is an implementation issue of the actual LFB, however this document defines operational parameters to control the splitting and merging, namely the size of the chunks, what happens if a packet or chunk has been marked as invalid and

whether the merge LFB should wait for all packets or chunks to arrive. Additionally this document defines metadata, which contain necessary information to assist the merging procedure. The following metadata set as a struct is defined:

1. ParallelType - Flood or split

2. Correlator - Identify packets or chunks that belonged to the initial packet that entered the Splitter LFB

3. ParallelNum - Number of packet or chunk for specific Correlator.

4. ParralelPartsCount - Total number of packets or chunks for specific Correlator.

This metadata is produced from the Splitter LFB and is opaque to LFBs in parallel paths and is passed along to the merger LFB without being consumed.

In case of a packet/chunk being branded invalid by an LFB in a parallel path, it MUST be sent by an output port of said LFB

An LFB inside a parallel path decides that a packet or a chunk has to be dropped it MAY drop it but the metadata MUST be sent to the Merger LFB's InvalidIn input port for merging purposes.

Additional metadata produced by LFBs inside a datapath MAY be aggregated within the Merger LFB and sent on after the merging process. In case of receiving the same metadata definition with multiple values the merger LFB MUST keep the first received from a valid packet or chunk.

4. Parallel Base Types

4.1. Frame Types

One frame type has been defined in this library.

Frame Type Name	Synopsis
Chunk	A chunk is a frame that is part of an original larger frame

Parallel Frame Types

4.2. Data Types

One data type has been defined in this library.

DataType Name	Type	Synopsis
ParallelTypes	Atomic uchar. Special Values Flood (0), Split (1).	The type of parallelization this packet will go through

Parallel Data Types

4.3. MetaData Types

The following metadata structure with ID 16, using the ForCES model extension [I-D.ietf-forces-model-extension], is defined for the parallelization library:

Metadata Name	Type	ID	Synopsis
ParallelType	uchar	1	The type of parallelization this packet will go through. 0 for flood, 1 for split.
Correlator	uint32	2	An identification number to specify that packets or chunks belong to the same parallel work.
ParallelNum	uint32	3	Defines the number of the specific packet or chunk of the specific parallel ID.
ParallelPartsCount	uint32	4	Defines the total number of packets or chunks for the specific parallel ID.

Metadata Structure for Merging

5. Parallel LFBs

5.1. Splitter

A splitter LFB takes part in parallelizing the processing datapath by sending either the same packet or chunks of the same packet to multiple LFBs.

5.1.1. Data Handling

The splitter LFB receives any kind of packet via the singleton input, Input. Depending upon the CE's configuration of the ParallelType component, if the parallel type is of type flood (0), the same packet MUST be sent through all of the group output ParallelOut's instances. If the parallel type is of type split (1), the packet will be split into same size chunks except the last which MAY be smaller, with the max size being defined by the ChunkSize component. All chunks will be sent out in a round-robin fashion through the group output ParallelOut's instances. Each packet or chunk will be accompanied by the following metadata set as a struct :

- ParallelType - The parallelltype split or flood.
- Parallel ID - generated by the splitter LFB to identify that chunks or packets belong to the same parallel work.
- Parallel Num - each chunk or packet of a parallel id will be assigned a number in order for the merger LFB to know when it has gathered them all along with the ParallelPartsCount metadata.
- ParallelPartsCount - the number of chunks or packets for the specific parallel id.

5.1.2. Components

This LFB has only two components specified. The first is the ParallelType, an uint32 that defines how the packet will be processed by the Splitter LFB. The second is the ChunkSize, an uint32 that specifies the maximum size of a chunk when a packet is split into multiple same size chunks.

5.1.3. Capabilities

This LFB has only one capability specified, the MinMaxChunkSize a struct of a uint32 to specify the minimum chunk size and a uint32 to specify the maximum chunk size.

5.1.4. Events

This LFB has no events specified.

5.2. Merger

A merger LFB receives multiple packets or multiple chunks of the same packet and merge them into one merged packet.

5.2.1. Data Handling

The Merger LFB receives either a packet or a chunk via the group input ParallelIn, along with the ParallelType metadata to identify whether what was received was a packet or a chunk, the Correlator, the ParallelNum and the ParallelPartsCount.

In case that an LFB has dropped a packet or a chunk within a parallel path the merger LFB MAY receive only the metadata or both metadata and packet or chunk through the InvalidIn group input port. It SHOULD receive a metadata specifying the error code. Current defined metadata's in the Base LFB Library [RFC6956] are the ExceptionID and the ValidateErrorID. The Merger LFB MAY store the parallel metadata along with the exception metadata as a string in the optional InvalideMetadataSets as a means for the CE to debug errors in the parallel path.

If the MergeWaitType is set to false the Merger LFB will initiate the merge process upon receiving the first packet. If false it will wait for all packet in the Correlator to arrive.

If one packet or chunk has been received through the InvalidIn port then the merging procedure will be operate as configured by the InvalidAction component. If the InvalidAction component has been set to 0 then if one packet or chunk is not valid all will dropped, else the process will initiate. Once the merging process has been finished the resulting packet will be sent via the singleton output port PacketOutput.

If the Merger LFB receives different values for the same metadata from different packets or chunks that has the same correlator then the Merger LFB will use the first metadata from a packet or chunk that entered the LFB through the ParallelIn input port.

5.2.2. Components

This LFB has the following components specified:

1. InvalidAction - a uchar defining what the Merge LFB will do if an invalid chunk or packet is received. If set to 0 (DropAll) the merge will be considered invalid and all chunks or packets will be dropped. If set to 1 (Continue) the merge will continue.

2. MergeWaitType - a boolean. If true the Merger LFB will wait for all packets or chunks to be received prior to sending out a response. If false, when one packet or a chunk with a response is received by the merge LFB it will start with the merge process.

3. InvalidMergesCounter - a uint32 that counts the number of merges where there is at least one packet or chunk that entered the merger LFB through the InvalidIn input port.

4. InvalidAllCounter - a uint 32 that counts the number of merges where all packets/chunks entered the merger LFB through the InvalidIn input port.

5. InvalidIDCounters - a struct of two arrays. Each array has a uint32 per row. Each array counts number of invalid merges where at least one packet or chunk entered through InvalidID per error ID. The first array is the InvalidExceptionID and the second is the InvalidValidateErrorID.

6. InvalideMetadataSets - an array of strings. An optional component that stores metadata sets along with the error id as a string. This could provide a debug information to the CE regarding errors in the parallel paths.

5.2.3. Capabilities

This LFB has no capabilities specified.

5.2.4. Events

This LFB specifies only two event. The first detects whether the InvalidMergesCounter has exceeded a specific value and the second detects whether the InvalidAllCounter has exceeded a specific value. Both error reports will send the respective counter value.

5.3. CoreParallelization

A core LFB that specifies that the FE supports parallelization, instead of updating the FEObject LFB

5.3.1. Data Handling

The CoreParallelization does not handle data. It is a core LFB that has only one capability.

5.3.2. Components

This LFB has no components specified.

5.3.3. Capabilities

This LFB has only one capability specified. The ParallelLFBs is a table which lists all the LFBs that can be parallelized. Each row of the table contains:

1. LFBName - a string. The Name of the parallel LFB.

2. LFBClassID - a uint32. The Class ID of the parallel LFB.

3. LFBVersion - a string. The Version of the parallel LFB.

4. LFBParallelOccurenceLimit - a uint32. The upper limit of instances of the same parallel LFBs of this class.

5. AllowedParallelAfters - a table of uint32s (LFB Class IDs). A list of LFB classes that can follow this LFB class in a pipeline for a parallel path.

6. AllowedParallelBefores - a table of uint32s (LFB Class IDs). A list of LFB classes that can exist before this LFB class in a pipeline for a parallel path.

7. AllowedParallel - a table of uint32s (LFB Class IDs). A list of LFB classes that can process packets or chunks in parallel with this LFB class.

5.3.4. Events

This LFB specifies no events

6. XML for Parallel 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"
   xsi:schemaLocation="urn:ietf:params:xml:ns:forces:lfbmodel:1.0"
   provides="Parallel">
   <load library="BaseTypeLibrary" location="BaseTypeLibrary.LFB" />
```

```
     <frameDefs>
        <frameDef>
           <name>Chunk</name>
           <synopsis>A chunk is a frame that is part of an original
              larger frame</synopsis>
        </frameDef>
     </frameDefs>
     <dataTypeDefs>
        <dataTypeDef>
           <name>ParallelTypes</name>
           <synopsis>The type of parallelization this packet will go
              through</synopsis>
           <atomic>
              <baseType>uchar</baseType>
              <specialValues>
                 <specialValue value="0">
                    <name>Flood</name>
                    <synopsis>The packet/chunk has been sent as a whole
                       to multiple recipients</synopsis>
                 </specialValue>
                 <specialValue value="1">
                    <name>Split</name>
                    <synopsis>The packet/chunk has been split into
                       multiple chunks and sent to recipients</synopsis>
                 </specialValue>
              </specialValues>
           </atomic>
        </dataTypeDef>
      <dataTypeDef>
         <name>ParallelLFBType</name>
         <synopsis>Table entry for parallel LFBs</synopsis>
         <struct>
            <component componentID="1">
               <name>LFBName</name>
               <synopsis>The name of an LFB Class</synopsis>
               <typeRef>string</typeRef>
            </component>
            <component componentID="2">
               <name>LFBClassID</name>
               <synopsis>The id of the LFB Class</synopsis>
               <typeRef>uint32</typeRef>
            </component>
            <component componentID="3">
               <name>LFBVersion</name>
               <synopsis>The version of the LFB Class used by this FE
               </synopsis>
               <typeRef>string</typeRef>
            </component>
```

```
            <component componentID="4">
               <name>LFBParallelOccurenceLimit</name>
               <synopsis>The upper limit of instances of the same
                  parallel LFBs of this class</synopsis>
               <optional />
               <typeRef>uint32</typeRef>
            </component>
            <component componentID="5">
               <name>AllowedParallelAfters</name>
               <synopsis>List of LFB Classes that can follow this LFB
                  in a parallel pipeline</synopsis>
               <optional />
               <array>
                  <typeRef>uint32</typeRef>
               </array>
            </component>
            <component componentID="6">
               <name>AllowedParallelBefores</name>
               <synopsis>List of LFB Classes that this LFB class can
                  follow in a parallel pipeline</synopsis>
               <optional />
               <array>
                  <typeRef>uint32</typeRef>
               </array>
            </component>
            <component componentID="7">
               <name>AllowedParallel</name>
               <synopsis>List of LFB Classes that this LFB class be run
                  in parallel with</synopsis>
               <array>
                  <typeRef>uint32</typeRef>
               </array>
            </component>
         </struct>
      </dataTypeDef>
   </dataTypeDefs>
   <metadataDefs>
      <metadataDef>
         <name>ParallelMetadataSet</name>
         <synopsis>A metadata Set for parallelization related LFBs
         </synopsis>
         <metadataID>32</metadataID>
         <struct>
            <component componentID="1">
               <name>ParallelType</name>
               <synopsis>The type of parallelization this packet/chunk
                  has gone through</synopsis>
               <typeRef>ParallelTypes</typeRef>
```

```
            </component>
            <component componentID="2">
               <name>Correlator</name>
               <synopsis>An identification number to specify that
                  packets or chunks originate from the same packet.
               </synopsis>
               <typeRef>uint32</typeRef>
            </component>
            <component componentID="3">
               <name>ParallelNum</name>
               <synopsis>Defines the number of the specific packet or
                  chunk of the specific parallel ID.</synopsis>
               <typeRef>uint32</typeRef>
            </component>
            <component componentID="4">
               <name>ParallelPartsCount</name>
               <synopsis>Defines the total number of packets or chunks
                  for the specific parallel ID.</synopsis>
               <typeRef>uint32</typeRef>
            </component>
         </struct>
      </metadataDef>
   </metadataDefs>
   <LFBClassDefs>
      <LFBClassDef LFBClassID="18">
         <name>Splitter</name>
         <synopsis>A splitter LFB takes part in parallelizing the
            processing datapath. It will either send the same packet
            or chunks of one packet to multiple LFBs</synopsis>
         <version>1.0</version>
         <inputPorts>
            <inputPort>
               <name>PacketIn</name>
               <synopsis>An input port expecting any kind of frame
               </synopsis>
               <expectation>
                  <frameExpected>
                     <ref>Arbitrary</ref>
                  </frameExpected>
               </expectation>
            </inputPort>
         </inputPorts>
         <outputPorts>
            <outputPort group="true">
               <name>ParallelOut</name>
               <synopsis>An parallel output port that sends the same
                  packet to all output instances or chunks of the same
                  packet different chunk on each instance.</synopsis>
```

```
                <product>
                   <frameProduced>
                      <ref>Arbitrary</ref>
                      <ref>Chunk</ref>
                   </frameProduced>
                   <metadataProduced>
                      <ref>ParallelMetadataSet</ref>
                   </metadataProduced>
                </product>
             </outputPort>
          </outputPorts>
          <components>
             <component componentID="1" access="read-write">
                <name>ParallelType</name>
                <synopsis>The type of parallelization this packet will
                   go through</synopsis>
                <typeRef>ParallelTypes</typeRef>
             </component>
             <component componentID="2" access="read-write">
                <name>ChunkSize</name>
                <synopsis>The size of a chunk when a packet is split
                   into multiple same size chunks</synopsis>
                <typeRef>uint32</typeRef>
             </component>
          </components>
          <capabilities>
             <capability componentID="31">
                <name>MinMaxChunkSize</name>
                <synopsis>The minimum and maximum size of a chunk
                   capable of splitted by this LFB</synopsis>
                <struct>
                   <component componentID="1">
                      <name>MinChunkSize</name>
                      <synopsis>Minimum chunk size</synopsis>
                      <optional/>
                      <typeRef>uint32</typeRef>
                   </component>
                   <component componentID="2">
                      <name>MaxChunkSize</name>
                      <synopsis>Maximum chunk size</synopsis>
                      <typeRef>uint32</typeRef>
                   </component>
                </struct>
             </capability>
          </capabilities>
       </LFBClassDef>
       <LFBClassDef LFBClassID="19">
          <name>Merger</name>
```

```
            <synopsis>A merger LFB receives multiple packets or multiple
               chunks of the same packet and merge them into one merged
               packet</synopsis>
            <version>1.0</version>
            <inputPorts>
               <inputPort group="true">
                  <name>ParallelIn</name>
                  <synopsis>An parallel input port that accepts packets
                     or chunks from all output instances</synopsis>
                  <expectation>
                     <frameExpected>
                        <ref>Arbitrary</ref>
                        <ref>Chunk</ref>
                     </frameExpected>
                     <metadataExpected>
                        <ref>ParallelMetadataSet</ref>
                     </metadataExpected>
                  </expectation>
               </inputPort>
               <inputPort group="true">
                  <name>InvalidIn</name>
                  <synopsis>When a packet is sent out of an error port of
                     an LFB in a parallel path will be sent to this
                     output port in the Merger LFB</synopsis>
                  <expectation>
                     <frameExpected>
                        <ref>Arbitrary</ref>
                        <ref>Chunk</ref>
                     </frameExpected>
                     <metadataExpected>
                        <one-of>
                           <ref>ExceptionID</ref>
                           <ref>ValidateErrorID</ref>
                        </one-of>
                     </metadataExpected>
                  </expectation>
               </inputPort>
            </inputPorts>
            <outputPorts>
               <outputPort>
                  <name>PacketOutput</name>
                  <synopsis>An output port expecting any kind of frame
                  </synopsis>
                  <product>
                     <frameProduced>
                        <ref>Arbitrary</ref>
                     </frameProduced>
                  </product>
```

```
            </outputPort>
         </outputPorts>
         <components>
            <component componentID="1" access="read-write">
               <name>InvalidAction</name>
               <synopsis>What the Merge LFB will do if an invalid
                  chunk or packet is received</synopsis>
               <atomic>
                  <baseType>uchar</baseType>
                  <specialValues>
                     <specialValue value="0">
                        <name>DropAll</name>
                        <synopsis>Drop all packets or chunks
                        </synopsis>
                     </specialValue>
                     <specialValue value="1">
                        <name>Continue</name>
                        <synopsis>Continue with the merge</synopsis>
                     </specialValue>
                  </specialValues>
               </atomic>
            </component>
            <component componentID="2" access="read-write">
               <name>MergeWaitType</name>
               <synopsis>Whether the Merge LFB will wait for all
                  packets or chunks to be received prior to sending
                  out a response</synopsis>
               <typeRef>boolean</typeRef>
            </component>
            <component componentID="3" access="read-reset">
               <name>InvalidMergesCounter</name>
               <synopsis>Counts the number of merges where there is at
                  least one packet/chunk that entered the merger LFB
                  through the InvalidIn input port</synopsis>
               <typeRef>uint32</typeRef>
            </component>
            <component componentID="4" access="read-reset">
               <name>InvalidAllCounter</name>
               <synopsis>Counts the number of merges where all
                  packets/chunks entered the merger LFB through the
                  InvalidIn input port</synopsis>
               <typeRef>uint32</typeRef>
            </component>
            <component componentID="5" access="read-reset">
               <name>InvalidIDCounters</name>
               <synopsis>Counts number of invalid merges where at
                  least one packet/chunk entered through InvalidID per
                  error ID</synopsis>
```

```
            <struct>
               <component componentID="1">
                  <name>InvalidExceptionID</name>
                  <synopsis>Per Exception ID</synopsis>
                  <array>
                     <typeRef>uint32</typeRef>
                  </array>
               </component>
               <component componentID="2">
                  <name>InvalidValidateErrorID</name>
                  <synopsis>Per Validate Error ID</synopsis>
                  <array>
                     <typeRef>uint32</typeRef>
                  </array>
               </component>
            </struct>
         </component>
         <component componentID="6" access="read-reset">
            <name>InvalideMetadataSets</name>
            <synopsis>Buffers metadata sets along with the error id
               as a string.</synopsis>
            <optional/>
            <array>
               <typeRef>string</typeRef>
            </array>
         </component>
      </components>
      <events baseID="30">
         <event eventID="1">
            <name>ManyInvalids</name>
            <synopsis>An event that specifies if there are too many
               invalids</synopsis>
            <eventTarget>
               <eventField>InvalidCounter</eventField>
            </eventTarget>
            <eventGreaterThan></eventGreaterThan>
            <eventReports>
               <eventReport>
                  <eventField>InvalidMergesCounter</eventField>
               </eventReport>
            </eventReports>
         </event>
         <event eventID="2">
            <name>ManyAllInvalids</name>
            <synopsis>An event that specifies if there are too many
               invalids</synopsis>
            <eventTarget>
               <eventField>InvalidCounter</eventField>
```

```
                </eventTarget>
                <eventGreaterThan></eventGreaterThan>
                <eventReports>
                   <eventReport>
                      <eventField>InvalidAllCounter</eventField>
                   </eventReport>
                </eventReports>
             </event>
          </events>
       </LFBClassDef>
       <LFBClassDef LFBClassID="20">
         <name>CoreParallelization</name>
         <synopsis>A core LFB that specifies that the FE supports
         parallelization, instead of updating the FEObject
         LFB</synopsis>
         <version>1.0</version>
         <capabilities>
            <capability componentID="10">
               <name>ParallelLFBs</name>
               <synopsis>A table which lists all the LFBs that can be
               parallelized</synopsis>
               <array>
                  <typeRef>ParallelLFBType</typeRef>
               </array>
            </capability>
         </capabilities>
       </LFBClassDef>
    </LFBClassDefs>
 </LFBLibrary>
```

Figure 4: Parallel LFB library

7. Acknowledgements

The authors would like to thank Jamal Hadi Salim and Dave Hood for comments and discussions that made this document better.

8. IANA Considerations

8.1. LFB Class Names and LFB Class Identifiers

LFB classes defined by this document belong to LFBs defined by Standards Track RFCs. According to IANA, the registration procedure is Standards Action for the range 0 to 65535 and First Come First Served with any publicly available specification for over 65535. This specification includes the following LFB class names and LFB class identifiers:

LFB Class Identifier	LFB Class Name	LFB Version	Description	Reference
18	Splitter	1.0	A splitter LFB will either send the same packet or chunks of one packet to multiple LFBs.	This document
19	Merger	1.0	A merger LFB receives multiple packets or multiple chunks of the same packet and merge them into one.	This document
20	CoreParallelization	1.0	A core LFB to signify the parallelization capability	This document

Logical Functional Block (LFB) Class Names and Class Identifiers

8.2. Metadata ID

The Metadata ID namespace is 32 bits long. Values assigned by this specification:

Value	Name	Definition
0x00000010	ParallelMetadataSet	This document

Metadata ID assigned by this specification

9. Security Considerations

10. References

10.1. Normative References

[I-D.ietf-forces-model-extension]
Haleplidis, E., "ForCES Model Extension", draft-ietf-forces-model-extension-01 (work in progress), November 2013.

[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.

[RFC6956] Wang, W., Haleplidis, E., Ogawa, K., Li, C., and J. Halpern, "Forwarding and Control Element Separation (ForCES) Logical Function Block (LFB) Library", RFC 6956, June 2013.

10.2. Informative References

[Cilc] MIT, "Cilk language", <http://supertech.csail.mit.edu/cilk/>.

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

Authors' Addresses

Evangelos Haleplidis
University of Patras
Department of Electrical and Computer Engineering
Patras 26500
Greece

Email: ehalep@ece.upatras.gr

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

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