

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: July 8, 2015

D. Joachimpillai
Verizon
J. Hadi Salim
Mojatatu Networks
January 4, 2015

**ForCES Inter-FE LFB
draft-ietf-forces-interfelfb-00**

Abstract

This document describes extending the ForCES LFB topology across FEs i.e inter-FE connectivity without needing any changes to the ForCES definitions by defining the Inter-FE LFB. The Inter-FE LFB provides ability to pass data, metadata and exceptions across FEs. The document describes a generic way to transport the mentioned details but focuses on ethernet transport.

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 July 8, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in [Section 4](#).e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Terminology and Conventions	3
1.1.	Requirements Language	3
1.2.	Definitions	3
2.	Introduction	3
3.	Problem Scope And Use Cases	4
3.1.	Basic Router	4
3.1.1.	Distributing The LFB Topology	6
3.2.	Arbitrary Network Function	7
3.2.1.	Distributing The Arbitrary Network Function	8
4.	Proposal Overview	9
4.1.	Inserting The Inter-FE LFB	9
5.	Generic Inter-FE connectivity	11
5.1.	Inter-FE Ethernet connectivity	13
5.1.1.	Inter-FE Ethernet Connectivity Issues	15
6.	Detailed Description of the Ethernet inter-FE LFB	16
6.1.	Data Handling	16
6.1.1.	Egress Processing	17
6.1.2.	Ingress Processing	18
6.2.	Components	19
6.3.	Inter-FE LFB XML Model	19
7.	Acknowledgements	24
8.	IANA Considerations	24
9.	Security Considerations	24
10.	References	25
10.1.	Normative References	25
10.2.	Informative References	25
	Authors' Addresses	25

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 reiterates the terminology defined in several ForCES documents [[RFC3746](#)], [[RFC5810](#)], [[RFC5811](#)], and [[RFC5812](#)] for the sake of contextual clarity.

Control Engine (CE)

Forwarding Engine (FE)

FE Model

LFB (Logical Functional Block) Class (or type)

LFB Instance

LFB Model

LFB Metadata

ForCES Component

LFB Component

ForCES Protocol Layer (ForCES PL)

ForCES Protocol Transport Mapping Layer (ForCES TML)

2. Introduction

In the ForCES architecture, a packet service can be modelled by composing a graph of one or more LFB instances. The reader is referred to the details in the ForCES Model [[RFC5812](#)].

The FEObject LFB capabilities in the ForCES Model [[RFC5812](#)] define component ModifiableLFBTopology which, when advertised by the FE, implies that the advertising FE is capable of allowing creation and modification the LFB graph by the control plane. Details on how a graph of LFB class instances can be created can be derived by the

control plane by looking at the FE's FEObject LFB class table component SupportedLFBs. The SupportedLFBs table contains information about each LFB class that the FE supports. For each LFB class supported, details are provided on how the supported LFB class may be connected to other LFB classes. The SupportedLFBs table describes which LFB class a specified LFB class may succeed or precede in an LFB class instance topology. Each link connecting two LFB class instances is described in the LFBLinkType dataTypeDef and has sufficient details to identify precisely the end points of a link of a service graph.

The CE may therefore create a packet service by describing an LFB instance graph connection; this is achieved by updating the FEObject LFBTopology table.

Often there are requirements for the packet service graph to cross FE boundaries. This could be from a desire to scale the service or need to interact with LFBs which reside in a separate FE (eg lookaside interface to a shared TCAM, an interconnected chip, or as coarse grained functionality as an external NAT FE box being part of the service graph etc).

Given that the ForCES inter-LFB architecture calls out for ability to pass metadata between LFBs, it is imperative therefore to define mechanisms to extend that existing feature and allow passing the metadata between LFBs across FEs.

This document describes extending the LFB topology across FEs i.e inter-FE connectivity without needing any changes to the ForCES definitions. It focusses on using Ethernet as the interconnection as a starting point while leaving room for other protocols (such as directly on top of IP, UDP, VXLAN, etc) to be addressed by other future documents.

3. Problem Scope And Use Cases

The scope of this document is to solve the challenge of passing ForCES defined metadata and exceptions across FEs (be they physical or virtual). To illustrate the problem scope we present two use cases where we start with a single FE running all the functionality then split it into multiple FEs.

3.1. Basic Router

A sample LFB topology Figure 1 demonstrates a service graph for delivering basic IPV4 forwarding service within one FE. For the purpose of illustration, the diagram shows LFB classes as graph nodes

instead of multiple LFB class instances.

Since the illustration is meant only as an exercise to showcase how data and metadata are sent down or upstream on a graph of LFBs, it abstracts out any ports in both directions and talks about a generic ingress and egress LFB. Again, for illustration purposes, the diagram does not show exception or error paths. Also left out are details on Reverse Path Filtering, ECMP, multicast handling etc. In other words, this is not meant to be a complete description of an IPV4 forwarding application; for a more complete example, please refer to the LFBlib document [[RFC6956](#)] .

The output of the ingress LFB(s) coming into the IPV4 Validator LFB will have both the IPV4 packets and, depending on the implementation, a variety of ingress metadata such as offsets into the different headers, any classification metadata, physical and virtual ports encountered, tunnelling information etc. These metadata are lumped together as "ingress metadata".

Once the IPV4 validator vets the packet (example ensures that no expired TTL etc), it feeds the packet and inherited metadata into the IPV4 unicast LPM LFB.

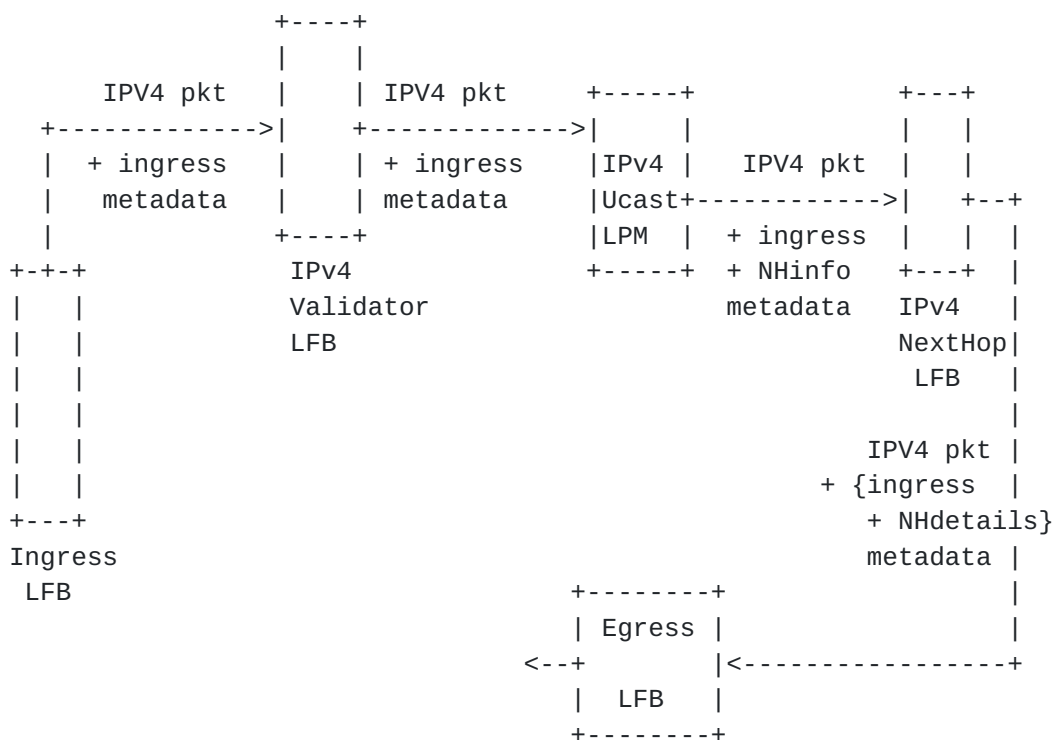


Figure 1: Basic IPV4 packet service LFB topology

The IPV4 unicast LPM LFB does a longest prefix match lookup on the IPV4 FIB using the destination IP address as a search key. The result is typically a next hop selector which is passed downstream as metadata.

The Nexthop LFB receives the IPv4 packet with an associated next hop info metadata. The NextHop LFB consumes the NH info metadata and derives from it a table index to look up the next hop table in order to find the appropriate egress information. The lookup result is used to build the next hop details to be used downstream on the egress. This information may include any source and destination information (MAC address to use, if ethernet;) as well egress ports. [Note: It is also at this LFB where typically the forwarding TTL decrement and IP checksum recalculation occurs.]

The details of the egress LFB are considered out of scope for this discussion. Suffice it is to say that somewhere within or beyond the Egress LFB the IPV4 packet will be sent out a port (ethernet, virtual or physical etc).

3.1.1. Distributing The LFB Topology

Figure 2 demonstrates one way the router LFB topology in Figure 1 may be split across two FEs (eg two ASICs). Figure 2 shows the LFB topology split across FEs after the IPV4 unicast LPM LFB.

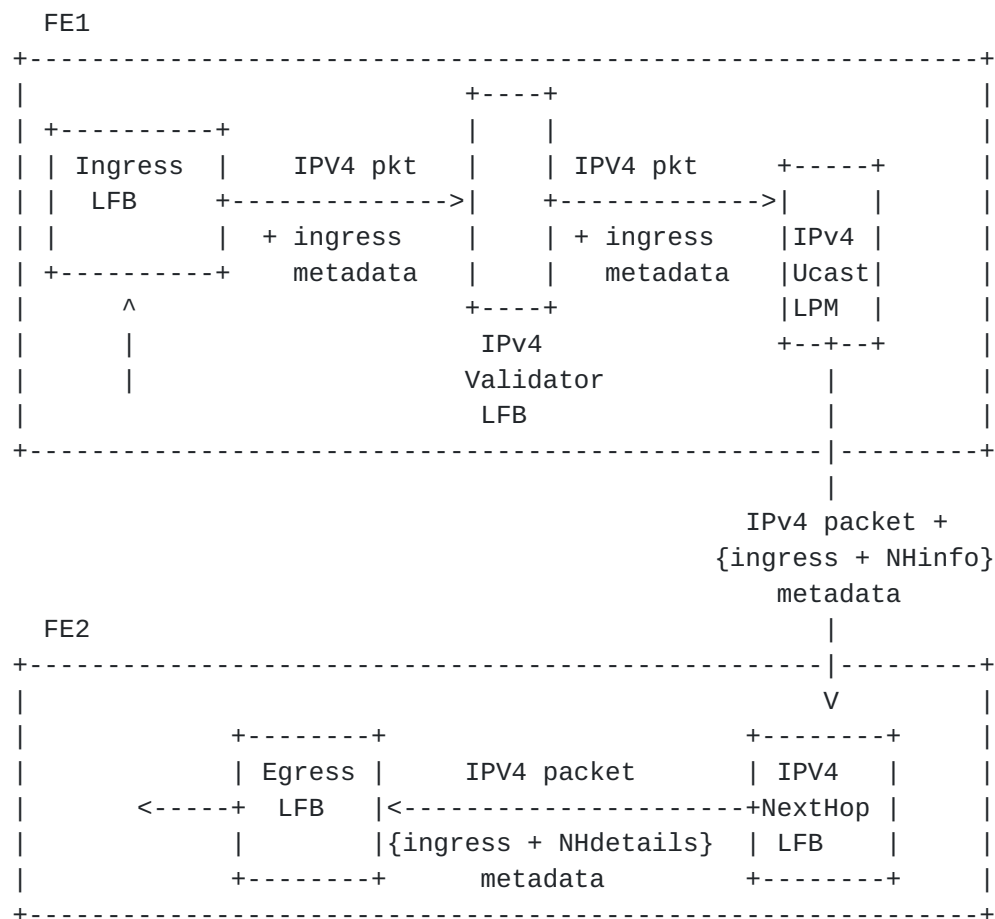


Figure 2: Split IPV4 packet service LFB topology

Some proprietary inter-connect (example Broadcom Higi over XAUI (XXX: ref needed)) maybe exist to carry both the IPV4 packet and the related metadata between the IPV4 Unicast LFB and IPV4 NextHop LFB across the two FEs.

The purpose of the inter-FE LFB is to define standard mechanisms for interconnecting FEs and for that reason we are not going to touch anymore on proprietary chip-chip interconnects other than state the fact they exist and that it is feasible to have translation to and from proprietary approaches. The document focus is the FE-FE interconnect where the FE could be physical or virtual and the interconnecting technology runs a standard protocol such as ethernet, IP or other protocols on top of IP.

3.2. Arbitrary Network Function

In this section we show an example of an arbitrary network function which is more coarse grained in terms of functionality. Each Network function may constitute more than one LFB.

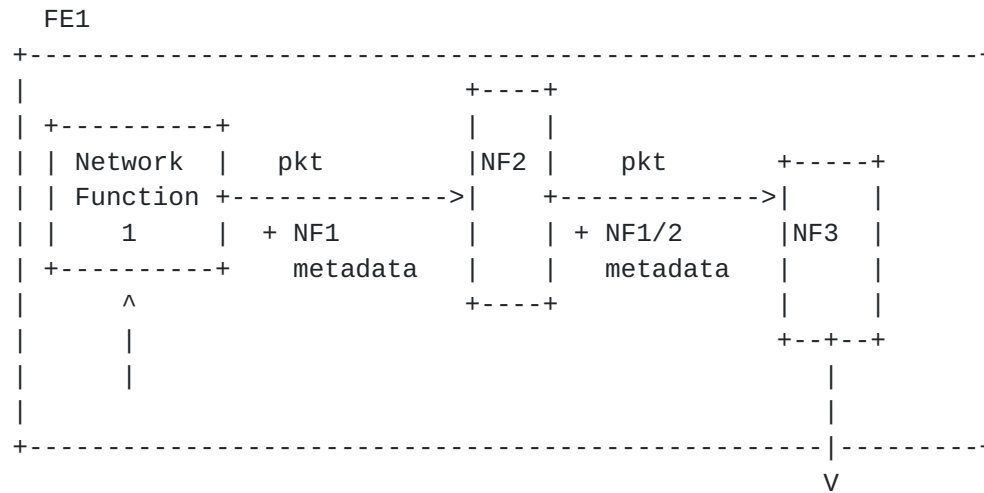


Figure 3: A Network Function Service Chain within one FE

The setup in Figure 3 is atypical of most packet processing boxes where we have functions like DPI, NAT, Routing, etc connected in such a topology to deliver a packet processing service to flows.

3.2.1. Distributing The Arbitrary Network Function

The setup in Figure 3 can be split out across 3 FEs instead as demonstrated in Figure 4. This could be motivated by scale out reasons or because different vendors provide different functionality which is plugged-in to provide such functionality. The end result is to have the same packet service delivered to the different flows passing through.

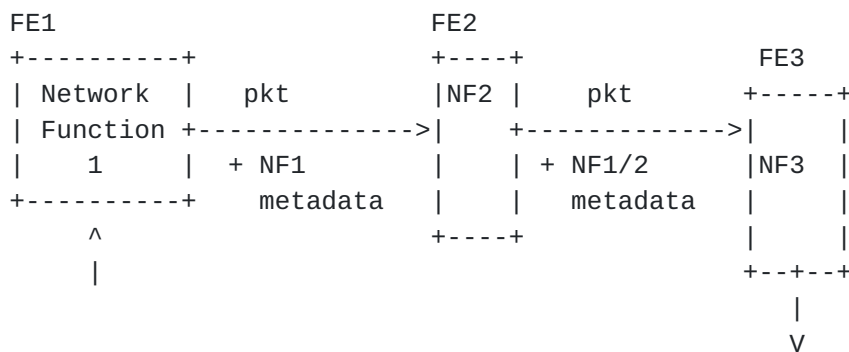


Figure 4: A Network Function Service Chain Distributed Across Multiple FEs

4. Proposal Overview

We address the inter-FE connectivity requirements by proposing the inter-FE LFB class. Using a standard LFB class definition implies no change to the basic ForCES architecture in the form of the core LFBs (FE Protocol or Object LFBs). This design choice was made after considering an alternative approach that would have required changes to both the FE Object capabilities (SupportedLFBs) as well LFBTopology component to describe the inter-FE connectivity capabilities as well as runtime topology of the LFB instances.

4.1. Inserting The Inter-FE LFB

The distributed LFB topology described in Figure 2 is re-illustrated in Figure 5 to show the topology location where the inter-FE LFB would fit in.

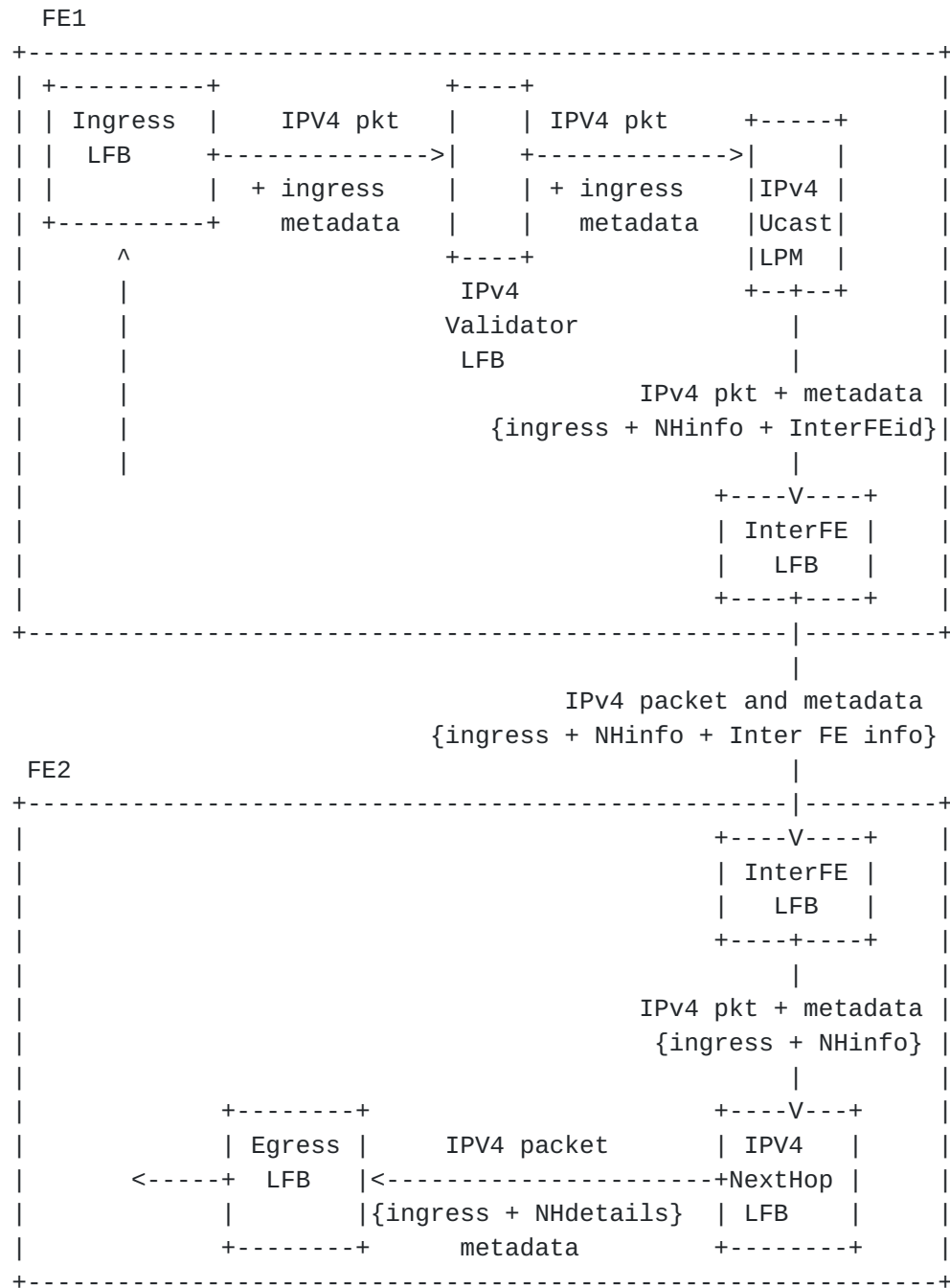


Figure 5: Split IPv4 forwarding service with Inter-FE LFB

As can be observed in Figure 5, the same details passed between IPv4 unicast LPM LFB and the IPv4 NH LFB are passed to the egress side of the Inter-FE LFB. In addition an index for the inter-FE LFB (interFEid) is passed as metadata.

The egress of the inter-FE LFB uses the received Inter-FE index (InterFEid metadata) to select details for encapsulation when sending

messages towards the selected neighboring FE. These details will include what to communicate as the source and destination FEID; in addition the original metadata and any exception IDs may be passed along with the original IPV4 packet.

On the ingress side of the inter-FE LFB the received packet and its associated details are used to decide the packet graph continuation. This includes what of the of the original metadata and exception IDs to restore and what next LFB class instance to continue processing on. In the illustrated case above, an IPV4 Nexthop LFB is selected and metadata is passed on to it.

The ingress side of the inter-FE LFB consumes some of the information passed (eg the destination FEID) and passes on the IPV4 packet alongside with the ingress + NHinfo metadata to the IPV4 NextHop LFB as was done earlier in both Figure 1 and Figure 2.

5. Generic Inter-FE connectivity

In this section we describe the generic encapsulation format in Figure 6 as extended from the ForCES redirect packet format. We intend for the described encapsulation to be a generic guideline of the different needed fields to be made available by any used transport for inter-FE LFB connectivity. We expect that for any transport mechanism used, a description of how the different fields will be encapsulated to be correlated to the information described in Figure 6. The goal of this document is to provide ethernet encapsulation, and to that end in [Section 5.1](#) we illustrate how we use the guidelines provided in this section to describe the fit for inter-FE LFB interfacing over ethernet.


```

+-- Main ForCES header
| |
| +---- msg type = REDIRECT
| +---- Destination FEID
| +---- Source FEID
| +---- NEID (first word of Correlator)
|
+-- T = ExceptionID-TLV
| |
| +-- +-Exception Data ILV (I = exceptionID , L= length)
| | | |
| | | +----- V= Metadata value
| | . |
| | . |
| | . +-Exception Data ILV
| | .
| |
+- T = METADATA-TLV
| |
| +-- +-Meta Data ILV (I = metaid, L= length)
| | | |
| | | +----- V= Metadata value
| | . |
| | . |
| | . +-Meta Data ILV
| | .
+- T = REDIRECTDATA-TLV
|
+-- Redirected packet Data

```

Figure 6: Packet format suggestion

- o The ForCES main header as described in [RFC5810](#) is used as a fixed header to describe the Inter-FE encapsulation.
- * The Source FEID field is mapped to the originating FE and the destination FEID is mapped to the destination FEID.
- * The first 32 bits of the correlator field are used to carry the NEID. The 32-bit NEID defaults to 0.
- o The ExceptionID TLV carries one or more exception IDs within ILVs. The I in the ILV carries a globally defined exceptionID as per-ForCES specification defined by IANA. This TLV is new to ForCES and sits in the global ForCES TLV namespace.
- o The METADATA and REDIRECTDATA TLV encapsulations are taken directly from [\[RFC5810\] section 7.9](#).

It is expected that a variety of transport encapsulations would be applicable to carry the format described in Figure 6. In such a case, a description of a mapping to interpret the inter-FE details and translate into proprietary or legacy formatting would need to be defined. For any mapping towards these definitions a different document to describe the mapping, one per transport, is expected to be defined.

5.1. Inter-FE Ethernet connectivity

In this document, we describe a format that is to be used over Ethernet.

The following describes the mapping from Figure 6 to ethernet wire encapsulation illustrated in Figure 7.

- o When an NE tag is needed, a VLAN tag will be used. Note: that the NEID as per Figure 6 is described as being 32 bits while a vlan tag is 12 bits. It is however thought to be sufficient to use 12 bits within the scope of a LAN NE cluster.
- o An ethernet type will be used to imply that a wire format is carrying an inter-FE LFB packet. The ethernet type will be requested from the appropriate IEEE Standards Association (preferred value is 0xFEFE because it maps nicely to the term FE-FE to imply inter-FE connectivity).
- o The destination FEID will be mapped to the destination MAC address of the target FEID.
- o The source FEID will be mapped to the source MAC address of the originating FEID.
- o In this version of the specification, we only focus on data and metadata. Therefore we are not going to describe how to carry the ExceptionID information (future versions may). We are also not going to use METADATA-TLV or REDIRECTDATA-TLV in order to save shave off some overhead bytes. Figure 7 describes the payload.

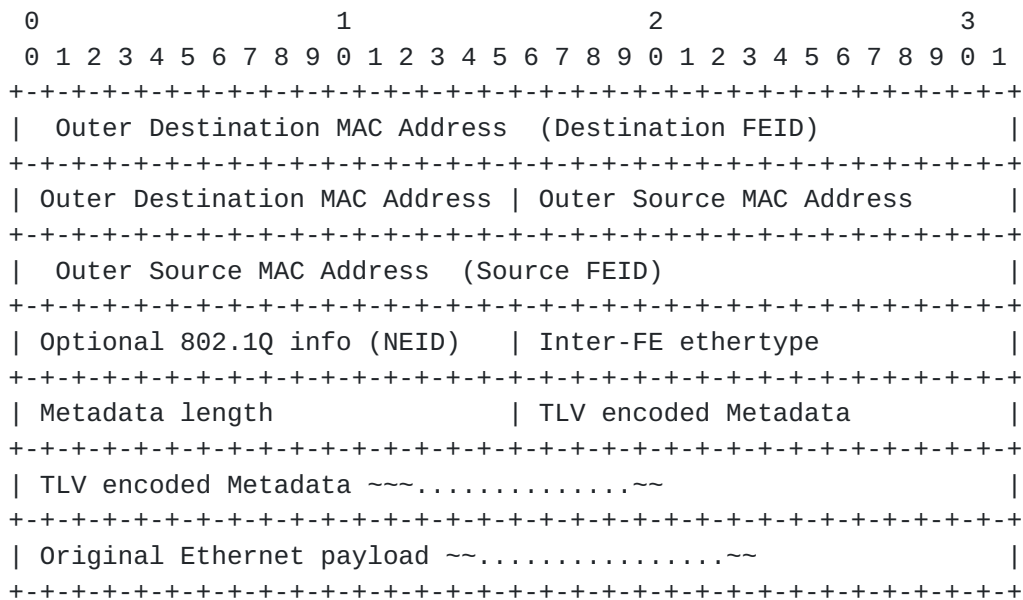


Figure 7: Packet format suggestion

An outer Ethernet header is introduced to carry the information on Destination FEID, Source FEID and optional NEID.

- o The Outer Destination MAC Address carries the Destination FEID identification.
- o Outer Source MAC Address carries the Source FEID identification.
- o When an NEID is needed, an optional 802.1Q is carried with 12-bit VLANid representing the NEID.
- o The ethernet type is used to identify the frame as inter-FE LFB type. Ethertype 0xFEFE is to be used (XXX: to be requested).
- o The 16-bit metadata length is used to described the total encoded metadata length (including the 16 bits used to encode the metadata length).
- o One or more TLV encoded metadatum follows the metadata length field. The TLV type identifies the Metadata id. ForCES IANA-defined Metadata ids will be used. We recognize that using a 16 bit TLV restricts the metadata id to 16 bits instead of ForCES define space of 32 bits. However, at the time of publication we believe this is sufficient to carry all the info we need and approach taken would save us 4 bytes per Metadatum transferred. XXX: If there is objection from the we could convert this to an ILV.

- o The original ethernet payload is appended at the end of the metadata as shown.

5.1.1. Inter-FE Ethernet Connectivity Issues

There are several issues that may arise due to using direct ethernet encapsulation.

- o Because we are adding data to existing ethernet frames, MTU issues may arise. We recommend:
 - * To use large MTUs when possible (example with jumbo frames).
 - * Limit the amount of metadata that could be transmitted; our definition allows for filtering of which metadata is to be encapsulated in the frame. We recommend complementing this by setting the egress port MTU to allow space for maximum size of the metadata total size you wish to allow between FEs. MTU setting can be achieved by configuration or ForCES control of the port LFB. In essence, the control plane making a decision for the MTU settings of the egress port is implicitly deciding how much metadata will be allowed.
- o The frame may be dropped if there is congestion on the receiving FE side. One approach to mitigate this issue is to make sure that inter-FE LFB frames receive the highest priority treatment when scheduled on the wire. Typically protocols that tunnel in the middle box do not care and depend on the packet originator to resend if the originator cares about reliability. We do not expect to be any different.
- o While we expect to use a unique IEEE-issued ethertype for the inter-FE traffic, we use lessons learnt from VXLAN deployment[XXX: ref] to be more flexible on the settings of the ethertype value used. We make the ethertype an LFB read-write component. Linux VXLAN implementation uses UDP port 8472 because the deployment happened much earlier than the point of RFC publication where the IANA assigned udp port issued was 4789. For this reason we make it possible to define at control time what ethertype to use and default to the IEEE issued ethertype. We justify this by assuming that a given ForCES NE is likely to be owned by a single organization and that the organization's CE(or CE cluster) could program all participating FEs via the inter-FE LFB (described in this document) to recognize a private ethernet type used for inter-LFB traffic (possibly those defined as available for private use by the IEEE, namely: IDs 0x88B5 and 0x88B6)

6. Detailed Description of the Ethernet inter-FE LFB

The ethernet inter-FE LFB has two LFB input ports and three LFB output ports.

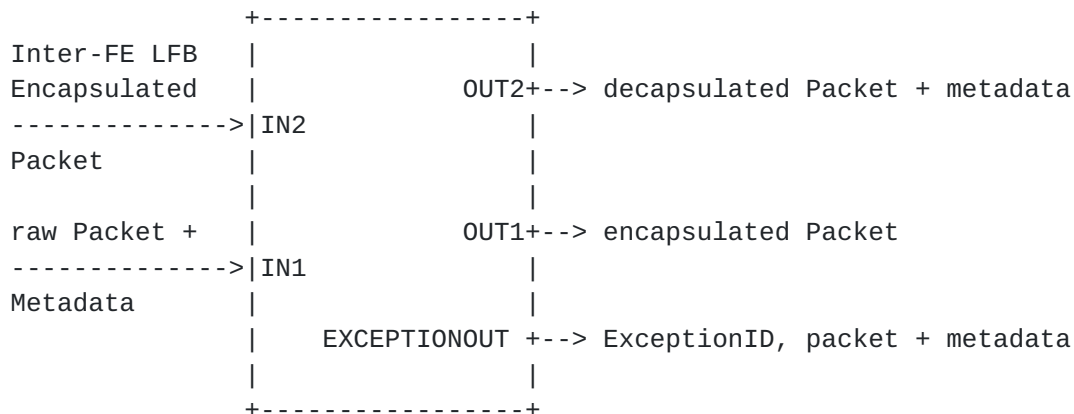


Figure 8: Inter-FE LFB

6.1. Data Handling

The Inter-FE LFB can be positioned at the egress of a source FE. In such a case an Inter-FE LFB instance receives via port IN1, raw packet and metadata IDs from the preceeding LFB instance. The InterFEid metadatum MAY be present on the incoming raw data. The processed encapsulated packet will go out on either LFB port OUT1 to a downstream LFB or EXCEPTIONOUT port in the case of a failure.

The Inter-FE LFB can be positioned at the ingress of a receiving FE. In such a case an Inter-FE LFB receives, via port IN2, an encapsulated packet. Successful processing of the packet will result in a raw packet with associated metadata IDs going downstream to an LFB connected on OUT2. On failure the data is sent out EXCEPTIONOUT.

The Inter-FE LFB may use the InterFEid metadatum on egress of an FE to lookup the IFETable table. The interFEid in such a case will be generated by an upstream LFB instance (i.e one preceeding the Inter-FE LFB). The output result constitutes a matched table row which has the InterFEinfo details i.e. the tuple {NEID, Destination FEID, Source FEID, inter FE type, metafilters}. The metafilters lists define which Metadatum are to be passed to the neighboring FE.

6.1.1. Egress Processing

The egress Inter-FE LFB will receive an ethernet frame and accompanying metadatum (including optionally the InterFEid metadatum) at LFB port IN1. The ethernet frame may be 802.1Q tagged.

The InterFEid may be used to lookup IFETable table. If lookup is successful, the inter-FE LFB will perform the following actions using the resulting tuple:

- o Increment statistics for packet and byte count observed.
- o Walk each passed metadatum apply against the MetaFilterList. If no legitimate metadata is found that needs to be passed downstream then the processing stops and the packet is allowed through as is.
- o check that the additional overhead of the outer header and encapsulated metadata will not exceed MTU. If it does increment the error packet count statistics and return allowing the packet to pass through. (XXX: Should it be dropped because it cannot send the required metadata?)
- o create the outer ethernet header which is a duplicate of the incoming frame's ethernet header. The outer ethernet header may have an optional 802.1q header (if one was included in the original frame).
- o If the NEID field is present (not 0) and the original header had a vlan tag, replace the vlan tag on the outer header with the value from the matched NEID field. If the NEID field is present (not 0) and the original header did not have a vlan tag, create one that matches the NEID field and appropriately add it to the outer header. If the NEID field is absent or 0, do nothing.
- o If the optional DSTFE is present, set the Destination MAC address of the outer header with value found in the DSTFE field. When absent, then the inner destination MAC address is used (at this point already copied).
- o If the optional SRCFE is present, set the Source MAC address of the outer header with value found in the SRCFE field. If SRCFE is absent then the inner source MAC address is used (at this point already copied).
- o If the optional IFETYPE is present, set the outer ethernet type to the value found in IFETYPE. If IFETYPE is absent then the standard ethernet type is used (XXX: to be requested from IEEE).

- o encapsulate each allowed metadatum in a TLV. Use the Metaid as the "type" field in the TLV header. The TLV should be aligned to 32 bits. This means you may need to add padding of zeroes to ensure alignment.
- o Update the Metadata length to the sum of each TLV's space + 2 bytes (for the Metadata length field 16 bit space).

The resulting packet is sent to the next LFB instance connected to the OUT1 LFB-port; typically a port LFB.

In the case of a failed lookup or a zero-value InterFEid, (or absence of InterFEid when needed by the implementation) the packet is sent out unchanged via the OUT1 LFB Class instance port (typically towards a Port LFB).

6.1.2. Ingress Processing

An inter-FE LFB packet is recognized by looking at the ethertype received on LFB instance port IN2. The IFETable table may be optionally utilized to provide metadata filters (XXX: Should we allow for the interFEid metadata to be sent by the neighbor FE and use it here? Implementations dont need it to associate a specific port/MAC/VLAN etc with the IFETable LFB).

- o Increment statistics for packet and byte count observed.
- o The inter-FE LFB instance looks at the metadata length field and walks the packet data extracting from the TLVs the metadata values. For each metadatum extracted, optionally the metaid is compared against the relevant IFETable row metafilter list. If the metadatum is recognized and is allowed by the filter the corresponding implementation metadatum field is set. If an unknown metadatum id is encountered, or if the metaid is not found in the option allowed filter list the implementation is expected to ignore it, increment the packet error statistic and proceed processing other metadatum.
- o Upon completion of processing all the metadata, the inter-FE LFB instance resets the header to point to the original (inner) ethernet header i.e skips the metadata information. At this point the the original ethernet frame that was passed to the egress Inter-FE LFB at the source FE is reconstructed. This data is then passed along with the reconstructed metadata downstream to the next LFB instance in the graph.

In the case of processing failure of either ingress or egress positioning of the LFB, the packet and metadata are sent out the

EXCEPTIONOUT LFB port with proper error id (XXX: More description to be added).

6.2. Components

There are two LFB component populated by the CE.

The CE optionally programs LFB instances in a service graph that require inter-FE connectivity with InterFEid values to correspond to the inter-FE LFB IFETable table entries to use.

The first component is an array known as the IFETable table. The array rows are made up of IFEInfo structure. The IFEInfo structure constitutes: optional NEID, optional IFETYPE, optional Destination FEID(DSTFE), optional Source FEID (SRCFE), optional array of allowed Metaids (MetaFilterList). The table is looked up by a 32 bit index passed from an upstream LFB class instance in the form of InterFEid metadatum.

The second component(ID 2) is IFESTats table which carries the basic stats structure bstats. The table index value used to lookup this table is the same one as in IFETable table; in other words for a table row index 10 in the IFETable table, its corresponding stats will be found in row index of the IFESTats table.

6.3. Inter-FE LFB XML Model

XXX: metadata definition requires clarification. We expect to use IANA defined metadatum. How do we describe them here in the model?

```
<LFBLibrary xmlns="urn:ietf:params:xml:ns:forces:lfbmodel:1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  provides="IFE">
  <frameDefs>

    <frameDef>
      <name>EthernetAny</name>
      <synopsis>Packet with any Ethernet type</synopsis>
    </frameDef>
    <frameDef>
      <name>InterFEFrame</name>
      <synopsis>
        Packet with an encapsulate IFE Ethernet type
      </synopsis>
    </frameDef>

  </frameDefs>
```



```
<dataTypeDefs>
```

```
  <dataTypeDef>
```

```
    <name>bstats</name>
```

```
    <synopsis>Basic stats</synopsis>
```

```
  <struct>
```

```
    <component componentID="1">
```

```
      <name>bytes</name>
```

```
      <synopsis>The total number of bytes seen</synopsis>
```

```
      <typeRef>uint64</typeRef>
```

```
    </component>
```

```
    <component componentID="2">
```

```
      <name>packets</name>
```

```
      <synopsis>The total number of packets seen</synopsis>
```

```
      <typeRef>uint32</typeRef>
```

```
    </component>
```

```
    <component componentID="3">
```

```
      <name>errors</name>
```

```
      <synopsis>The total number of packets with errors</synopsis>
```

```
      <typeRef>uint32</typeRef>
```

```
    </component>
```

```
  </struct>
```

```
</dataTypeDef>
```

```
  <dataTypeDef>
```

```
    <name>IFEInfo</name>
```

```
    <synopsis>Describing IFE table row Information</synopsis>
```

```
  <struct>
```

```
    <component componentID="1">
```

```
      <name>NEID</name>
```

```
      <synopsis>
```

```
        The VLAN Id 12 bits part of the 802.1q TCI field.
```

```
      </synopsis>
```

```
      <optional/>
```

```
      <typeRef>uint16</typeRef>
```

```
    </component>
```

```
    <component componentID="2">
```

```
      <name>IFETYPE</name>
```

```
      <synopsis>
```

```
        the ethernet type to be used for outgoing IFE frame
```

```
      </synopsis>
```

```
      <optional/>
```

```
      <typeRef>uint16</typeRef>
```

```
    </component>
```

```
    <component componentID="3">
```



```
<name>DSTFE</name>
<synopsis>
    the destination MAC address of destination FE
</synopsis>
<optional/>
<typeRef>byte[6]</typeRef>
</component>
<component componentID="4">
    <name>SRCFE</name>
    <synopsis>
        the source MAC address used for the source FE
    </synopsis>
    <optional/>
    <typeRef>byte[6]</typeRef>
</component>
<component componentID="5">
    <name>MetaFilterList</name>
    <synopsis>
        the metadata filter table
    </synopsis>
    <optional/>
    <array type="variable-size">
        <typeRef>uint32</typeRef>
    </array>
</component>

</struct>
</dataTypeDef>

</dataTypeDefs>

<metadataDefs>
    <metadataDef>
        <name>InterFEid</name>
        <synopsis>
            Metadata identifying the index of the NexFE table
        </synopsis>
        <metadataID>16</metadataID>
        <typeRef>uint32</typeRef>
    </metadataDef>
</metadataDefs>

<LFBClassDefs>
    <LFBClassDef LFBClassID="6612">
        <name>IFE</name>
        <synopsis>
            This LFB describes IFE connectivity parametrization
```



```
</synopsis>
<version>1.0</version>

<inputPorts>

  <inputPort>
    <name>IN1</name>
    <synopsis>
      The input port of the egress side.
      It expects any type of Ethernet frame.
    </synopsis>
    <expectation>
      <frameExpected>
        <ref>EthernetAny</ref>
      </frameExpected>
    </expectation>
  </inputPort>
  <inputPort>
    <name>IN2</name>
    <synopsis>
      The input port of the ingress side.
      It expects an inter-FE encapsulated Ethernet frame
      with associated metadata.
    </synopsis>
    <expectation>
      <frameExpected>
        <ref>InterFEFrame</ref>
      </frameExpected>
      <metadataExpected>
        <ref>InterFEid</ref>
      </metadataExpected>
    </expectation>
  </inputPort>
</inputPorts>

<outputPorts>

  <outputPort>
    <name>OUT1</name>
    <synopsis>
      The output port of the egress side.
    </synopsis>
    <product>
      <frameProduced>
        <ref>InterFEFrame</ref>
      </frameProduced>
      <metadataProduced>
```



```
        <ref>InterFEid</ref>
      </metadataProduced>
    </product>
  </outputPort>

  <outputPort>
    <name>OUT2</name>
    <synopsis>
      The output port of the Ingress side.
    </synopsis>
    <product>
      <frameProduced>
        <ref>EthernetAny</ref>
      </frameProduced>
      <metadataProduced>
        <ref>InterFEid</ref>
      </metadataProduced>
    </product>
  </outputPort>

  <outputPort>
    <name>EXCEPTIONOUT</name>
    <synopsis>
      The exception handling path
    </synopsis>
    <product>
      <frameProduced>
        <ref>EthernetAny</ref>
      </frameProduced>
      <metadataProduced>
        <ref>ExceptionID</ref>
        <ref>InterFEid</ref>
      </metadataProduced>
    </product>
  </outputPort>

</outputPorts>

<components>

  <component componentID="1" access="read-write">
    <name>IFETable</name>
    <synopsis>
      the table of all InterFE relations
    </synopsis>
    <array type="variable-size">
      <typeRef>IFEInfo</typeRef>
    </array>
```



```
</component>
<component componentID="2">
  <name>IFEStats</name>
  <synopsis>
    the stats corresponding to the IFETable table
  </synopsis>
  <typeRef>bstats</typeRef>
</component>

</components>

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

Figure 9: Inter-FE LFB XML

7. Acknowledgements

The authors would like to thank Joel Halpern and Dave Hood for the stimulating discussions. Evangelos Haleplidis contributed to improving this document.

8. IANA Considerations

This memo includes two IANA requests within the registry <https://www.iana.org/assignments/forces>

The first request is for the sub-registry "Logical Functional Block (LFB) Class Names and Class Identifiers" to request for the reservation of LFB class name IFE with LFB classid 6112 with version 1.0.

The second request is for the sub-registry "Metadata ID" to request for the InterFEid metadata the value 0x00000010.

9. Security Considerations

XXX:TBD

10. References

10.1. Normative References

- [RFC3746] Yang, L., Dantu, R., Anderson, T., and R. Gopal, "Forwarding and Control Element Separation (ForCES) Framework", [RFC 3746](#), April 2004.
- [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.
- [RFC5811] Hadi Salim, J. and K. Ogawa, "SCTP-Based Transport Mapping Layer (TML) for the Forwarding and Control Element Separation (ForCES) Protocol", [RFC 5811](#), March 2010.
- [RFC5812] Halpern, J. and J. Hadi Salim, "Forwarding and Control Element Separation (ForCES) Forwarding Element Model", [RFC 5812](#), March 2010.

10.2. Informative References

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

Authors' Addresses

Damascene M. Joachimpillai
Verizon
60 Sylvan Rd
Waltham, Mass. 02451
USA

Email: damascene.joachimpillai@verizon.com

Jamal Hadi Salim
Mojatatu Networks
Suite 400, 303 Moodie Dr.
Ottawa, Ontario K2H 9R4
Canada

Email: hadi@mojatatu.com

