Networking Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 7, 2016

Ran. Chen
Zheng. Zhang
ZTE Corporation
Vengada. Govindan
IJsbrand. Wijnands
Cisco

March 12, 2016

# BGP Link-State extensions for BIER
# draft-chenvgovindan-bier-bgp-ls-bier-ext-00

## Abstract

Bit Index Explicit Replication (BIER) is an architecture that provides optimal multicast forwarding through a "BIER domain" without requiring intermediate routers to maintain any multicast related per-flow state. BIER also does not require any explicit tree-building protocol for its operation. A multicast data packet enters a BIER domain at a "Bit-Forwarding Ingress Router" (BFIR), and leaves the BIER domain at one or more "Bit-Forwarding Egress Routers" (BFERs). The BFIR router adds a BIER header to the packet. The BIER header contains a bitstring in which each bit represents exactly one BFER to forward the packet to. The set of BFERs to which the multicast packet needs to be forwarded is expressed by setting the bits that correspond to those routers in the BIER header.

This document specifies extensions to the BGP Link-state address-family in order to advertising BIER information.

## 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 7, 2016.

Table of Contents



## 1. Introduction

Bit Index Explicit Replication (BIER) is an architecture that provides optimal multicast forwarding through a "BIER domain" without requiring intermediate routers to maintain any multicast related per-flow state. BIER also does not require any explicit tree-building protocol for its operation. A multicast data packet enters a BIER domain at a "Bit-Forwarding Ingress Router" (BFIR), and leaves the BIER domain at one or more "Bit-Forwarding Egress Routers" (BFERs). The BFIR router adds a BIER header to the packet. The BIER header contains a bitstring in which each bit represents exactly one BFER to forward the packet to. The set of BFERs to which the multicast packet needs to be forwarded is expressed by setting the bits that correspond to those routers in the BIER header.

This document specifies extensions to the BGP Link-state address-family in order to advertising BIER-specific. An external component (e.g., a controller) then can collect BIER information in the "northbound" direction within the BIER domain.

## 2. Conventions used in this document

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.

## 3. BGP-LS Extensions for BIER

Each BFR MUST be assigned a "BFR-Prefix". A BFR's BFR-Prefix MUST be an IP address (either IPv4 or IPv6) of the BFR, and MUST be unique and routable within the BIER domain as described in section 2 of [I-D.ietf-bier-architecture], and then external component (e.g., a controller) need to collect BIER information of BIER routers are associated with the BFR-Prefix in the "northbound" direction within the BIER domain.

Given that the BIER information is associated with the prefix, the BGP-LS Prefix Attribute TLV [I-D.ietf-idr-ls-distribution] can be used to carry the BIER information. A new Prefix Attribute TLV and Sub-TLV are defined for the encoding of BIER information.

### 3.1. The BIER TLV

A new Prefix Attribute TLV (defined in [I-D.ietf-idr-ls-distribution] is defined for distributing BIER information. The new TLV is called the BIER TLV. The BIER TLVs may appear multiple times.

The following BIER TLV is defined:

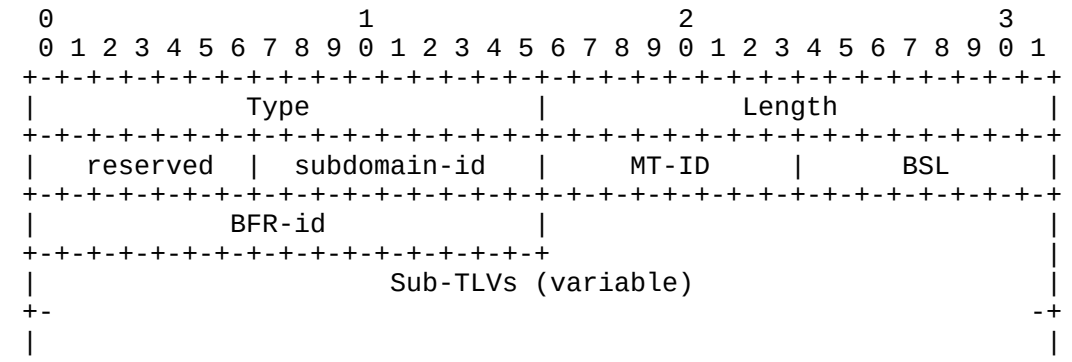



Figure 1

Type: TBD.

Length: 2 octet.

Subdomain-id: Unique value identifying the BIER sub-domain, 1 octet.

MT-ID: Multi-Topology ID that identifies the topology that is associated with the BIER sub-domain.1 octet.

BitString Length (BS Len): A 1 octet field encoding the supported BitString length associated with this BFR-prefix.This field are specified in section 3 of [I-D.ietf-bier-architecture].Given that the bier router can support BSL values set, this field encoding the BSL values set that BIER routers supported.

BFR-id: A 2 octet field encoding the BFR-id, as documented in [I-D.ietf-bier-architecture]. If the BFR-id is zero, it means, the advertising router is not advertising any BIER-id.

If multiple BIER Sub-TLVs are present, all having the same BS Length and Subdomain-id values, first one MUST be used and subsequent ones MUST be ignored.

### 3.1.1. The BIER MPLS Encapsulation Sub-TLV

The BIER MPLS Encapsulation Sub-TLV is a sub-TLV of the BIER TLV. BIER MPLS Encapsulation Sub-TLV is used in order to advertise MPLS specific information used for BIER. It MUST appear multiple times in the BIER TLV as described in [I-D.ietf-bier-ospf-bier-extensions]

The following the BIER MPLS Encapsulation Sub-TLV is defined:

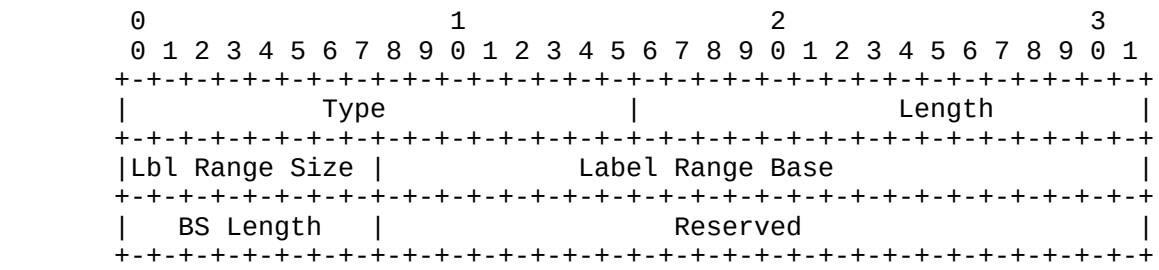



Figure 2

Type: TBD.

Length: 2 octet.

Label Range Size: A 1 octet field encoding the label range size of the label range. It MUST be greater than 0, otherwise the TLV MUST be ignored.

Label Range Base: A 3 octet field, where the 20 rightmost bits represent the first label in the label range.

BS Length: Bitstring length for the label range that this router is advertising per [I-D.ietf-bier-mpls-encapsulation]. 1 octet.The values allowed in this field are specified in section 3 of [I-D.ietf-bier-mpls-encapsulation].

The "label range" is the set of labels beginning with the label range base and ending with (label range base)+(label range size)-1. A unique label range is allocated for each BitStream length and Sub-domain-ID. These label is used for BIER forwarding as described in [I-D.ietf-bier-architecture] and [I-D.ietf-bier-mpls-encapsulation].Label ranges within the sub-TLV MUST NOT overlap, otherwise the whole sub-TLV MUST be disregarded

BS length in multiple BIER MPLS Encapsulation Sub-TLV inside the same BIER Sub-TLV MUST NOT repeat, otherwise only the first BIER MPLS Encapsulation Sub-TLV with such BS length MUST be used and any subsequent BIER MPLS Encapsulation Sub-TLVs with the same BS length MUST be ignored.

## 3.2. The BIER-TE TLV

This TLV is used to collect BIER-TE information in the "northbound" direction within the BIER-TE domain.

The section will be added in next version.

# 4. IANA Considerations

This document requests assigning code-points from the registry for the new Prefix Attribute TLV and Sub-TLV.

| TLV Code Point | Description | Value defined |
|---|---|---|
| 1158( recommend ) | BIER | this document |

Table 1: The new Prefix Attribute TLV

| Sub-TLV Code Point | Description | Value |
|---|---|---|
| 1 ( recommend) | BIER MPLS Encapsulation | this document |

Table 2: The new Prefix Attribute Sub-TLV

## 5. Security Considerations

Procedures and protocol extensions defined in this document do not affect the BGP security model. See [RFC6952] for details.

## 6. Acknowledgements

We would like to thank Peter Psenak (Cisco) for his comments and support of this work.

## 7. Normative references

[I-D.ietf-bier-architecture]
Wijnands, I., Rosen, E., Dolganow, A., P, T., and S. Aldrin, "Multicast using Bit Index Explicit Replication", draft-ietf-bier-architecture-03 (work in progress), January 2016.

[I-D.ietf-bier-isis-extensions]
Ginsberg, L., P, T., Aldrin, S., and J. Zhang, "BIER support via ISIS", draft-ietf-bier-isis-extensions-01 (work in progress), October 2015.

[I-D.ietf-bier-mpls-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., and S. Aldrin, "Encapsulation for Bit Index Explicit Replication in MPLS Networks", draft-ietf-bier-mpls-encapsulation-03 (work in progress), February 2016.

[I-D.ietf-bier-ospf-bier-extensions]
Psenak, P., Kumar, N., Wijnands, I., Dolganow, A., P, T., Zhang, J., and S. Aldrin, "OSPF Extensions For BIER", draft-ietf-bier-ospf-bier-extensions-01 (work in progress), October 2015.

[I-D.ietf-idr-ls-distribution]
Gredler, H., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and TE Information using BGP", draft-ietf-idr-ls-distribution-13 (work in progress), October 2015.

[RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <http://www.rfc-editor.org/info/rfc3630>.

[RFC6952] Jethanandani, M., Patel, K., and L. Zheng, "Analysis of BGP, LDP, PCEP, and MSDP Issues According to the Keying and Authentication for Routing Protocols (KARP) Design Guide", RFC 6952, DOI 10.17487/RFC6952, May 2013, <http://www.rfc-editor.org/info/rfc6952>.

## Authors' Addresses

Ran Chen
ZTE Corporation
No.50 Software Avenue,Yuhuatai District
Nanjing, Jiangsu Province 210012
China

Phone: +86 025 88014636
Email: chen.ran@zte.com.cn

Zheng Zhang
ZTE Corporation
No.50 Software Avenue,Yuhuatai District
Nanjing, Jiangsu Province 210012
China

Email: zhang.zheng@zte.com.cn

Vengada Prasad Govindan
Cisco

Email: venggovi@cisco.com

IJsbrand Wijnands
Cisco
De Kleetlaan 6a
Diegem 1831
Belgium

Email: ice@cisco.com

BIER WG
Internet-Draft
Intended status: Standards Track
Expires: September 19, 2016

Ran. Chen
Fangwei. Hu
Zheng. Zhang
Xianxia. Dai
ZTE Corporation
Mahesh Sivakumar
Cisco Systems, Inc.
March 18, 2016

# YANG Data Model for BIER Protocol
draft-chh-bier-bier-yang-03.txt

## Abstract

This document defines a YANG data model for BIER configuration and operation.

## 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 19, 2016.

## Copyright Notice

Table of Contents



## 1. Introduction

This document defines a YANG data model for BIER configuration and operation.

## 2. Design of the Data Model

```
module: ietf-bier
augment /rt:routing:
   +--rw bier
      +--rw bier-global
         +--rw encapsulation-type?   identityref
         +--rw bitstringlength?      bsl
         +--rw bfr-id?               bfr-id
         +--rw ipv4-bfr-prefix?   inet:ipv4-prefix
         +--rw ipv6-bfr-prefix?   inet:ipv6-prefix
         +--rw sub-domain* [sub-domain-id]
            +--rw sub-domain-id      sub-domain-id
                        +--rw igp-type?          igp-type
            +--rw mt-id?             mt-id
            +--rw bfr-id?            bfr-id
            +--rw frr?               boolean
            +--rw bitstringlength?   bsl
            +--rw af
               +--rw ipv4* [bitstringlength bier-mpls-label-base]
               |  +--rw bitstringlength               uint16
               |  +--rw bier-mpls-label-base          mpls:mpls-label
               |  +--rw bier-mpls-label-range-size?   bier-mpls-label-range-size
```

```
               +--rw ipv6* [bitstringlength bier-mpls-label-base]
                  +--rw bitstringlength              uint16
                  +--rw bier-mpls-label-base         mpls:mpls-label
                  +--rw bier-mpls-label-range-size?   bier-mpls-label-range-size
```

```
  augment /rt:routing/rt:routing-instance/rt:routing-protocols
         /rt:routing-protocol/ospf:ospf/ospf:instance:
   +--rw bier-ospf-cfg
      +--rw mt-id          mt-id
      +--rw bier-global
         +--rw enable?      boolean
         +--rw advertise?   boolean
         +--rw receive?     boolean

   augment /rt:routing/rt:routing-instance/rt:routing-protocols
          /rt:routing-protocol/isis:isis/isis:instance:
   +--rw bier-isis-cfg
      +--rw mt-id          mt-id
      +--rw bier-global
         +--rw enable?      boolean
         +--rw advertise?   boolean
         +--rw receive?     boolean
```

```
  augment /rt:routing-state:
   +--ro bier-global
   |  +--ro encapsulation-type?   identityref
   |  +--ro bitstringlength?      bsl
   |  +--ro bfr-id?               bfr-id
   |  +--ro ipv4-bfr-prefix?      inet:ipv4-prefix
   |  +--ro ipv6-bfr-prefix?      inet:ipv6-prefix
   |  +--ro sub-domain* [sub-domain-id]
   |     +--ro sub-domain-id    sub-domain-id
   |     +--ro igp-type         igp-type
   |     +--ro mt-id?           mt-id
   |     +--ro bfr-id?          bfr-id
   |     +--ro frr?             boolean
   |     +--rw bitstringlength?   bsl
   |     +--ro ipv4* [bitstringlength bier-mpls-label-base]
   |     |  +--ro bitstringlength               uint16
   |     |  +--ro bier-mpls-label-base          mpls:mpls-label
   |     |  +--ro bier-mpls-label-range-size?   bier-mpls-label-range-size
   |     +--ro ipv6* [bitstringlength bier-mpls-label-base]
   |        +--ro bitstringlength               uint16
   |        +--ro bier-mpls-label-base          mpls:mpls-label
   |        +--ro bier-mpls-label-range-size?   bier-mpls-label-range-size
   +--ro birts
```

```
      +--ro birt* [sub-domain-id]
         +--ro sub-domain-id           sub-domain-id
         +--ro birt-bitstringlength* [bitstringlength]
            +--ro bitstringlength    uint16
            +--ro birt-si* [si]
               +--ro si                     si
               +--ro f-bm?                  uint16
               +--ro bier-mpls-in-label?    mpls:mpls-label
               +--ro bfr-nbr?               inet:ip-address
               +--ro bier-mpls-out-label?   mpls:mpls-label
notifications:
   +---n bfr-id-collision
   |  +--ro bfr-id?   bfr-id
   +---n bfr-zero
   |  +--ro ipv4-bfr-prefix?   inet:ipv4-prefix
   |  +--ro ipv6-bfr-prefix?   inet:ipv6-prefix
   +---n sub-domain-id-collision
      +--ro sub-domain-id?   sub-domain-id
      +--ro mt-id?           uint16
```

## 3. Configuration

This Module augments the "/rt:routing:" with a BIER container. This Container defines all the configuration parameters related to BIER for this particular routing.

The BIER configuration contains global configuration.

The global configuration includes BIER encapsulation type, imposition BitStringLengths, BFR-id, BFR-prefixes, and parameters associated with bier sub-domain.

In this document, we contains two types of BitStringLengths: Imposition and Disposition BitStringLengths, as defined in ([I-D.ietf-bier-architecture]).The imposition BitStringLengths is defined under bier-global container, and the disposition BitStringLengths is defined under the sub-domain.

## 4. Control plane configuration

This Module augments the "/rt:routing/rt:routing-instance/rt:routing-protocolsbier-protocol-extensions /rt:routing-protocol/ospf:ospf/ ospf:instance:" and "/rt:routing/rt:routing-instance/rt:routing-protocols /rt:routing-protocol/isis:isis/isis:instance:"configuration with BIER.

This Module supports ISIS ([I-D.ietf-bier-isis-extensions])and OSPF ([I-D.ietf-bier-ospf-bier-extensions]) as control plane for BIER.

## 5. States

The operational states contains basic parameters associated with bier, such as BIER encapsulation type, BitStringLengths, BFR-id, BFR-prefixes, and parameters associated with bier sub-domain.

It also includes the Bit Index Routing Table(BIRT).

## 6. Notification

This Module includes bfr-id-collision, bfr-zero, and sub-domain-id-collision.

## 7. BIER YANG Data Model

```
<CODE BEGINS> file "ietf-bier.yang"
module ietf-bier {

     namespace "urn:ietf:params:xml:ns:yang:ietf-bier";

     prefix "bier";

     import ietf-routing {
       prefix "rt";
     }

     import ietf-inet-types {
       prefix "inet";
     }

         import ietf-mpls {
            prefix "mpls";
          }

         import ietf-isis{
       prefix "isis";
     }

     import ietf-ospf  {
       prefix "ospf";
     }

     organization
       "IETF BIER(Bit Indexed Explicit Replication ) Working Group";
```

```
     contact
        "WG List:  <mailto:bier@ietf.org>

        WG Chair: Tony Przygienda
                  <mailto:tonysietf@gmail.com>

        WG Chair: Greg Shepherd
                  <mailto:gjshep@gmail.com>


        Editor:   Ran Chen
                   <mailto:chen.ran@zte.com.cn>
        Editor:   Fangwei Hu
                   <mailto:hu.fangwei@zte.com.cn>
        Editor:   Zheng Zhang
                   <mailto:zhang.zheng@zte.com.cn>
        Editor:   Xianxian Dai
                   <mailto:dai.xianxian@zte.com.cn>
        Editor:    Mahesh Sivakumar
                   <mailto:masivaku@cisco.com>
        ";
       description
         "The YANG module defines a generic configuration
          model for BIER.";

       revision 2016-03-16 {
         description
           "04 revision";
         reference
           "draft-chh-bier-bier-yang-03";
       }

       revision 2015-12-03 {
         description
           "03 revision";
         reference
           "draft-chh-bier-bier-yang-02";
       }

       revision 2015-11-19 {
         description
           "02 revision, typedef the parameters related with bier, change the ty
pe of label to mpls-label";
         reference
           "draft-chh-bier-bier-yang-01";
       }

       revision 2015-10-16 {
         description
```

```
          "01 revision.";
        reference
          "draft-chh-bier-bier-yang-01";
      }

      revision 2015-06-22 {
        description
          "Initial revision.";
        reference
          "draft-chh-bier-bier-yang-00";
    }

    /* Identities */
    identity bier-encapsulation{
      description
        "Base identity for BIER encapsulation.";
    }
    identity bier-encapsulation-mpls {
      base bier-encapsulation;
      description
        "This identity represents MPLS encapsulation for bier.";
    }

       /*Typedefs*/

       typedef sub-domain-id {
         type uint16;
         description
           "The type for sub-domain-id";
        }

       typedef si {
         type uint16;
         description
           "The type for set identifier";
        }

        typedef bfr-id {
          type uint16;
          description
            "The type for bfr identifier";
        }

        typedef mt-id {
          type uint16;
          description
            "The type for multi-topology identifier";
        }
```

```
        typedef bier-mpls-label-range-size{
          type uint8;
          description
                "The type for BIER label range size.";
               }

        typedef bsl {
          type enumeration{
                enum 64-bit{
                  value 1;
                  description
                    "bitstringlength is 64";
                      }
                enum 128-bit{
                  value 2;
                  description
                        "bitstringlength is 128";
                      }
                enum 256-bit{
                  value 3;
                  description
                    "bitstringlength is 256";
                      }
                enum 512-bit{
                  value 4;
                        description
                          "bitstringlength is 512";
                      }
                enum 1024-bit{
                  value 5;
                  description
                    "bitstringlength is 1024";
                      }
                enum 2048-bit{
                  value 6;
                  description
                    "bitstringlength is 2048";
                      }
                enum 4096-bit{
                  value 7;
                  description
                    "bitstringlength is 4096";
                      }
               }
                description
                  "The bitstringlength type for imposition mode";
       }
```

```
         typedef igp-type {
           type enumeration{
                 enum ISIS{
                   value 1;
                   description
                     "isis protocol";
                        }
                 enum OSPF{
                   value 2;
                   description
                     "ospf protocol";
                        }
                  }
         description
                   "The IGP type";
                 }

        /*grouping*/
          grouping bier-protocol-extensions{
                leaf mt-id{
                  type mt-id;
                  description
                        "Multi-topology associated with bier sub-domain.";
                 }
             container bier-global {
                   leaf enable {
                     type boolean;
                         default false;
                         description
                           "Enables bier protocol extensions.";
                   }
                   leaf advertise {
                         type boolean;
                         default true;
                         description
                           "Enable to advertise the parameters associated with bi
er.";
                         }
                   leaf receive {
                         type boolean;
                         default true;
                         description
                           "Enable to receive the parameters associated with bier
.";
                         }
                   description
                         "BIER global config.";
                }
            description
                  "Defines protocol extensions.";
```

```
           }

          grouping bier-parameters{
                leaf encapsulation-type {
                  type identityref {
                    base bier-encapsulation;
                        }
                  default "bier-encapsulation-mpls";
                  description
                        "Dataplane to be used.";
                }
                leaf bitstringlength{
                  type bsl;
                  description
                    "imposition bitstringlength.";
                }
                leaf bfr-id {
                  type bfr-id;
                  description
                        "BIER bfr identifier.";
                }
                leaf ipv4-bfr-prefix {
                  type inet:ipv4-prefix;
                  description
                        "BIER IPv4 prefix.";
                }
                leaf ipv6-bfr-prefix {
                  type  inet:ipv6-prefix;
                  description
                        "BIER IPv6 prefix.";
                }
                list sub-domain{
        key "sub-domain-id";
                leaf sub-domain-id{
                  type sub-domain-id;
                  description
                        "sub-domain ID.";
                }
                leaf igp-type {
                  type igp-type;
                  description
                        "IGP type.";
                }
                leaf mt-id {
                  type mt-id;
                  description
                        "multi-topology ID.";
                }
```

```
                leaf bfr-id{
                  type bfr-id;
                  description
                        "BIER bfr identifier.";
                }
                leaf frr{
                  type boolean;
                  description
                        "Enables BIER FRR.";
                }
          leaf bitstringlength{
                                type bsl;
                                  description
                                        "Disposition bitstringlength.";
                               }
            description
          "Denfines subdomain configuration";
             }
            description
              " BIER parameters.";
            }

            grouping bier-mpls-cfg{
                  leaf bitstringlength {
                    type uint16;
                    description
                          "BIER bitstringlength.";
                  }
                  leaf bier-mpls-label-base{
                    type mpls:mpls-label;
                    description
                          "BIER label base.";
                  }
                  leaf bier-mpls-label-range-size{
                    type bier-mpls-label-range-size;
                    description
                          "BIER label range.";
                  }
            description
                  "Defines the necessary label ranges per bitstring length.";
            }

       /* Configuration Data */
        augment "/rt:routing" {
              description
                    "This augments routing-instance configuration with bier.";
              container bier{
                    container bier-global {
```

```
                    uses bier-parameters;
                          container af {
                                list ipv4 {
                                  key "bitstringlength bier-mpls-label-base";
                                  uses bier-mpls-cfg;

                                  description
                                        "Defines the necessary label ranges per
bitstring length in ipv4.";
                                        }
                                list ipv6 {
                                  key "bitstringlength bier-mpls-label-base";
                                  uses bier-mpls-cfg;

                                  description
                                    "Defines the necessary label ranges per bits
tring length in ipv6.";
                                        }
                             description
                                   "Bier mapping entries.";
                                   }
                  description
                    "BIER global config.";
                  }
               description "BIER config.";
             }
          }

     augment "/rt:routing/rt:routing-instance/rt:routing-protocols/"
           + "rt:routing-protocol/ospf:ospf/ospf:instance" {
       when "../../rt:type = 'ospf:ospfv2' or
             ../../rt:type = 'ospf:ospfv3'" {
         description
           "This augments the ospf routing protocol when used";
         }
       description
         "This augments ospf protocol configuration with bier.";
                container bier-ospf-cfg{
                  uses bier-protocol-extensions;
           description
                        "Control of bier advertisement and reception.";
                }
      }

     augment "/rt:routing/rt:routing-instance/" +
           "rt:routing-protocols/rt:routing-protocol"+
           "/isis:isis" {
       when "rt:type = 'isis:isis'" {
         description
         "This augment ISIS routing protocol when used";
```

```
       }
       description
         "This augments ISIS protocol configuration with bier.";
                  container bier-isis-cfg{
            uses bier-protocol-extensions;
                        description
                          "Control of bier advertisement and reception.";

                  }
      }

         /* Operational  data */
      augment "/rt:routing-state" {
        description
                  "This augments the operational states with bier.";
                container bier-global{
                        uses bier-parameters;
                        list ipv4 {
                          key "bitstringlength bier-mpls-label-base";
                          uses bier-mpls-cfg;
                            description
                                  "Show the necessary label ranges per bitstring
 length in ipv4.";
                                }
                        list ipv6 {
                          key "bitstringlength bier-mpls-label-base";
                                uses bier-mpls-cfg;
                                  description
                                        "Show the necessary label ranges per bit
string length in ipv6.";
                                }
             description
                          "Parameters associated with bier.";
                        }

                container birts{
                  list birt{
                        key "sub-domain-id";
                        leaf sub-domain-id{
                          type sub-domain-id;
                                description
                                  "BIER sub domain ID";
                                }
                        list birt-bitstringlength {
                          key "bitstringlength";
                          leaf bitstringlength{
                                type uint16;
                                  description
                                        "BIER bitstringlength.";
                             }
```

```
                        list birt-si {
                              key "si";
                              leaf si{
                                type si;
                                description
                                      "BIER set identifier.";
                                      }
                              leaf f-bm{
                                type uint16;
                                description
                                      "BIER Forwarding Bit Mask.";
                                             }
                              leaf bier-mpls-in-label{
                                type mpls:mpls-label;
                                description
                                      "BIER in-label.";
                                             }
                              leaf bfr-nbr{
                                type inet:ip-address;
                                description
                                "BIER BFR Neighbors.";
                                             }
                              leaf bier-mpls-out-label{
                                type mpls:mpls-label;
                                description
                                      "BIER out-label.";
                                             }
                          description
                                "Query the BIRT based on the key set identifier
 & bitstringlength & sub-domain-id.";
                              }
                       description
                         "Query the BIRT based on the key bitstringlengt & sub-
domain-id.";
                              }
                    description
                          "Query the BIRT based on the key sub-domain.";
             }
                  description
                        "Shows Bit Index Routing Table.";
                  }
           }


          /* Notifications */
      notification bfr-id-collision{
                leaf bfr-id{
                  type bfr-id;
                  description
                        "BIER BFR ID.";
```

```
        }
           description
                 "BFR ID received in the controlplane that caused BFR ID collisi
on.";
       }

      notification bfr-zero{
                leaf ipv4-bfr-prefix{
                  type inet:ipv4-prefix;
                  description
                        "BIER ipv4 bfr prefix";
         }
                leaf ipv6-bfr-prefix{
                  type inet:ipv6-prefix;
                  description
                        "BIER ipv6 bfr prefix";
        }
                description
                        "Invalid value associated with prefix";
       }

      notification sub-domain-id-collision{
                leaf sub-domain-id{
                  type sub-domain-id;
                  description
                        "BIER sub domain ID";
        }
                leaf mt-id{
                  type uint16;
                  description
                        "Multi-topology ID";
        }
           description
                 "Sub domain ID received in the controlplane that caused Sub dom
ain ID collision";
       }
     }
```

<CODE ENDS>

## 8. Security Considerations

TBD.

## 9. Acknowledgements

We would like to thank IJsbrand Wijnands, Reshad Rahman and Giles Heron for their comments and support of this work.

## 10. IANA Considerations

This document requires no IANA Actions. Please remove this section before RFC publication.

## 11. References

### 11.1. Normative references

[I-D.ietf-bier-architecture]
Wijnands, I., Rosen, E., Dolganow, A., P, T., and S. Aldrin, "Multicast using Bit Index Explicit Replication", draft-ietf-bier-architecture-03 (work in progress), January 2016.

[I-D.ietf-bier-isis-extensions]
Ginsberg, L., P, T., Aldrin, S., and J. Zhang, "BIER support via ISIS", draft-ietf-bier-isis-extensions-01 (work in progress), October 2015.

[I-D.ietf-bier-mpls-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., and S. Aldrin, "Encapsulation for Bit Index Explicit Replication in MPLS Networks", draft-ietf-bier-mpls-encapsulation-03 (work in progress), February 2016.

[I-D.ietf-bier-ospf-bier-extensions]
Psenak, P., Kumar, N., Wijnands, I., Dolganow, A., P, T., Zhang, J., and S. Aldrin, "OSPF Extensions For BIER", draft-ietf-bier-ospf-bier-extensions-01 (work in progress), October 2015.

[I-D.ietf-isis-yang-isis-cfg]
Litkowski, S., Yeung, D., Lindem, A., Zhang, J., and L. Lhotka, "YANG Data Model for IS-IS protocol", draft-ietf-isis-yang-isis-cfg-07 (work in progress), November 2015.

[I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-20 (work in progress), October 2015.

[I-D.ietf-ospf-yang]
Yeung, D., Qu, Y., Zhang, J., Bogdanovic, D., and K. Koushik, "Yang Data Model for OSPF Protocol", draft-ietf-ospf-yang-03 (work in progress), October 2015.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <http://www.rfc-editor.org/info/rfc6991>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

### 11.2. Informative references

[I-D.saad-mpls-static-yang] Raza, K., Gandhi, R., Liu, X., Beeram, V., Saad, T., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for MPLS Base and Static LSPs", draft-saad-mpls-static-yang-01 (work in progress), December 2015.

## Authors' Addresses

Ran Chen
ZTE Corporation
No.50 Software Avenue,Yuhuatai District
Nanjing, Jiangsu Province 210012
China

Phone: +86 025 88014636
Email: chen.ran@zte.com.cn

Fangwei Hu
ZTE Corporation
No.889 Bibo Rd
Shanghai 201203
China

Phone: +86 21 68896273
Email: hu.fangwei@zte.com.cn

Zheng Zhang
ZTE Corporation
No.50 Software Avenue,Yuhuatai District
Nanjing, Jiangsu Province 210012
China

Email: zhang.zheng@zte.com.cn

Xianxian Dai
ZTE Corporation
No.50 Software Avenue,Yuhuatai District
Nanjing, Jiangsu Province 210012
China

Email: Dai.xianxian@zte.com.cn

Mahesh Sivakumar
Cisco Systems, Inc.
510 McCarthy Blvd
Milpitas,California 95035
United States

Email: masivaku@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 21, 2016

T. Eckert
Cisco Systems, Inc.
G. Cauchie
Bouygues Telecom
W. Braun
M. Menth
University of Tuebingen
March 20, 2016

# Traffic Engineering for Bit Index Explicit Replication BIER-TE
draft-eckert-bier-te-arch-03

## Abstract

This document proposes an architecture for BIER-TE: Traffic Engineering for Bit Index Explicit Replication (BIER).

BIER-TE shares part of its architecture with BIER as described in [I-D.ietf-bier-architecture]. It also proposes to share the packet format with BIER.

BIER-TE forwards and replicates packets like BIER based on a BitString in the packet header but it does not require an IGP. It does support traffic engineering by explicit hop-by-hop forwarding and loose hop forwarding of packets. It does support Fast ReRoute (FRR) for link and node protection and incremental deployment. Because BIER-TE like BIER operates without explicit in-network tree-building but also supports traffic engineering, it is more similar to SR than RSVP-TE.

Table of Contents

# 1. Introduction

## 1.1. Overview

This document specifies the architecture for BIER-TE: traffic engineering for Bit Index Explicit Replication BIER.

BIER-TE shares architecture and packet formats with BIER as described in [I-D.ietf-bier-architecture].

BIER-TE forwards and replicates packets like BIER based on a BitString in the packet header but it does not require an IGP. It does support traffic engineering by explicit hop-by-hop forwarding and loose hop forwarding of packets. It does support Fast ReRoute (FRR) for link and node protection and incremental deployment. Because BIER-TE like BIER operates without explicit in-network tree-

building but also supports traffic engineering, it is more similar to Segment Routing (SR) than RSVP-TE.

The key differences over BIER are:

- BIER-TE replaces in-network autonomous path calculation by explicit paths calculated offpath by the BIER-TE controller host.
- In BIER-TE every BitPosition of the BitString of a BIER-TE packet indicates one or more adjacencies - instead of a BFER as in BIER.
- BIER-TE in each BFR has no routing table but only a BIER-TE Forwarding Table (BIFT) indexed by SI:BitPosition and populated with only those adjacencies to which the BFR should replicate packets to.

BIER-TE headers use the same format as BIER headers.

BIER-TE forwarding does not require/use the BFIR-ID. The BFIR-ID can still be useful though for coordinated BFIR/BFER functions, such as the context for upstream assigned labels for MPLS payloads in MVPN over BIER-TE.

If the BIER-TE domain is also running BIER, then the BFIR-ID in BIER-TE packets can be set to the same BFIR-ID as used with BIER packets.

If the BIER-TE domain is not running full BIER or does not want to reduce the need to allocate bits in BIER bitstrings for BFIR-ID values, then the allocation of BFIR-ID values in BIER-TE packets can be done through other mechanisms outside the scope of this document, as long as this is appropriately agreed upon between all BFIR/BFER.

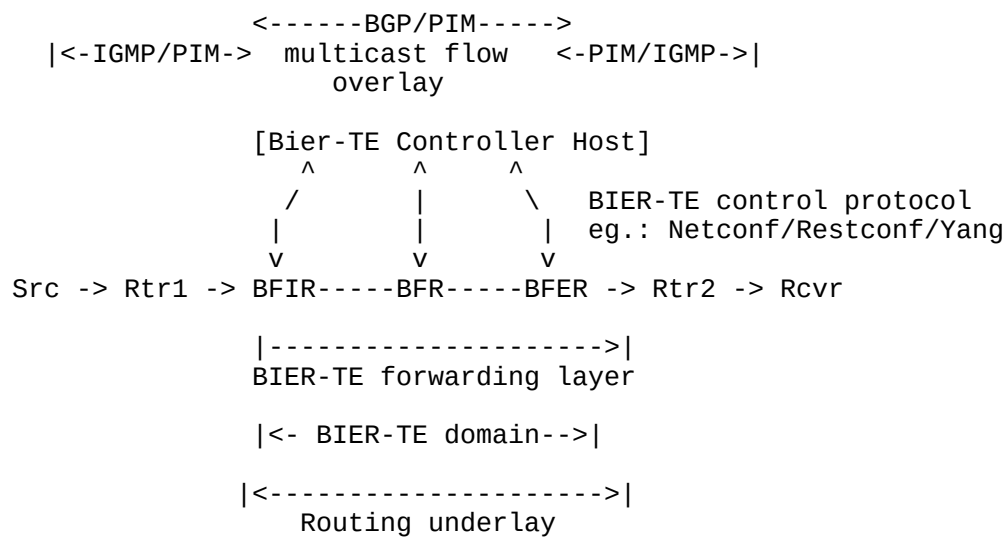
### 1.2. Requirements Language

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 RFC 2119 [RFC2119].

## 2. Layering

End to end BIER-TE operations consists of four components: The "Multicast Flow Overlay", the "BIER-TE Controller Host", the "Routing Underlay" and the "BIER-TE forwarding layer".

Picture 2: Layers of BIER-TE




## 2.1. The Multicast Flow Overlay

The Multicast Flow Overlay operates as in BIER. See [I-D.ietf-bier-architecture]. Instead of interacting with the BIER layer, it interacts with the BIER-TE Controller Host

## 2.2. The BIER-TE Controller Host

The BIER-TE controller host is representing the control plane of BIER-TE. It communicates two sets of information with BFRs:

During bring-up or modifications of the network topology, the controller discovers the network topology, assigns BitPositions to adjacencies and signals the resulting mapping of BitPositions to adjacencies to each BFR connecting to the adjacency.

During day-to-day operations of the network, the controller signals to BFIRs what multicast flows are mapped to what BitStrings.

Communications between the BIER-TE controller host to BFRs is ideally via standardized protocols and data-models such as Netconf/Retconf/ Yang. This is currently outside the scope of this document. Vendor-specific CLI on the BFRs is also a possible stopgap option (as in many other SDN solutions lacking definition of standardized data model).

For simplicity, the procedures of the BIER-TE controller host are described in this document as if it is a single, centralized automated entity, such as an SDN controller. It could equally be an operator setting up CLI on the BFRs. Distribution of the functions of the BIER-TE controller host is currently outside the scope of this document.

#### 2.2.1. Assignment of BitPositions to adjacencies of the network topology

The BIER-TE controller host tracks the BFR topology of the BIER-TE domain. It determines what adjacencies require BitPositions so that BIER-TE explicit paths can be built through them as desired by operator policy.

The controller then pushes the BitPositions/adjacencies to the BIFT of the BFRs, populating only those SI:BitPositions to the BIFT of each BFR to which that BFR should be able to send packets to - adjacencies connecting to this BFR.

#### 2.2.2. Changes in the network topology

If the network topology changes (not failure based) so that adjacencies that are assigned to BitPositions are no longer needed, the controller can re-use those BitPositions for new adjacencies. First, these BitPositions need to be removed from any BFIR flow state and BFR BIFT state (and BTAFT if FRR is supported, see below), then they can be repopulated, first into BIFT (and if FRR is supported BTAFT), then into BFIR.

#### 2.2.3. Set up per-multicast flow BIER-TE state

The BIER-TE controller host tracks the multicast flow overlay to determine what multicast flow needs to be sent by a BFIR to which set of BFER. It calculates the desired distribution tree across the BIER-TE domain based on algorithms outside the scope of this document (eg.: CSFP, Steiner Tree,...). It then pushes the calculated BitString into the BFIR.

#### 2.2.4. Link/Node Failures and Recovery

When link or nodes fail or recover in the topology, BIER-TE can quickly respond with the optional FRR procedures described below. It can also more slowly react by recalculating the BitStrings of affected multicast flows. This reaction is slower than the FRR procedure because the controller needs to receive link/node up/down indications, recalculate the desired BitStrings and push them down

into the BFIRs. With FRR, this is all performed locally on a BFR receiving the adjacency up/down notification.

### 2.3. The BIER-TE Forwarding Layer

When the BIER-TE Forwarding Layer receives a packet, it simply looks up the BitPositions that are set in the BitString of the packet in the Bit Index Forwarding Table (BIFT) that was populated by the BIER-TE controller host. For every BP that is set in the BitString, and that has one or more adjacencies in the BIFT, a copy is made according to the type of adjacencies for that BP in the BIFT. Before sending any copy, the BFR resets all BitPositions in the BitString of the packet to which it can create a copy. This is done to inhibit that packets can loop.

If the BFR support BIER-TE FRR operations, then the BIER-TE forwarding layer will receive fast adjacency up/down notifications. The BIER-TE FRR Adjacency Table uses the notifications to modify the BIER header and the bitstring of the packet before it performs BIER-TE forwarding. This is detailed in the FRR section.

### 2.4. The Routing Underlay

BIER-TE is sending BIER packets to directly connected BIER-TE neighbors as L2 (unicasted) BIER packets without requiring a routing underlay. BIER-TE forwarding uses the Routing underlay for forward_routed adjacencies which copy BIER-TE packets to not-directly-connected BFRs (see below for adjacency definitions).

If the BFR intends to support FRR for BIER-TE, then the BIER-TE forwarding plane needs to receive fast adjacency up/down notifications: Link up/down or neighbor up/down, eg.: from BFD. Providing these notifications is considered to be part of the routing underlay in this document.

## 3. BIER-TE Forwarding

### 3.1. The Bit Index Forwarding Table (BIFT)

The Bit Index Forwarding Table (BIFT) exists in every BFR. For every subdomain in use, it is a table indexed by SI:BitPosition and is populated by the BIER-TE control plane. Each index can be empty or contain a list of one or more adjacencies.

BIER-TE can support multiple subdomains like BIER. Each one with a separate BIFT

In the BIER architecture, indices into the BIFT are explained to be both BFR-id and SI:BitString (BitPosition). This is because there is a 1:1 relationship between BFR-id and SI:BitString - every bit in every SI is/can be assigned to a BFIR/BFER. In BIER-TE there are more bits used in each BitString than there are BFIR/BFER assigned to the bitstring. This is because of the bits required to express the (traffic engineered) path through the topology. The BIER-TE forwarding definitions do therefore not use the term BFR-id at all. Instead, BFR-ids are only used as required by routing underlay, flow overlay of BIER headers. Please refer to Section 8 for explanations how to deal with SI, subdomains and BFR-id in BIER-TE.

| Index: SI:BitPosition | Adjacencies: <empty> or one or more per entry |
|---|---|
| 0:1 | forward_connected(interface,neighbor,DNR) |
| 0:2 | forward_connected(interface,neighbor,DNR)<br>forward_connected(interface,neighbor,DNR) |
| 0:3 | local_decap([VRF]) |
| 0:4 | forward_routed([VRF,]l3-neighbor) |
| 0:5 | <empty> |
| 0:6 | ECMP({adjacency1,...adjacencyN}, seed) |
| ... | |
| BitStringLength | ... |

Bit Index Forwarding Table

The BIFT is programmed into the data plane of BFRs by the BIER-TE controller host and used to forward packets, according to the rules specified in the BIER-TE Forwarding Procedures.

Adjacencies for the same BP when populated in more than one BFR by the controller do not have to have the same adjacencies. This is up to the controller. BPs for p2p links are one case (see below).

## 3.2. Adjacency Types

#### 3.2.1. Forward Connected

A "forward_connected" adjacency is towards a directly connected BFR neighbor using an interface address of that BFR on the connecting interface. A forward_connected adjacency does not route packets but only L2 forwards them to the neighbor.

Packets sent to an adjacency with "DoNotReset" (DNR) set in the BIFT will not have the BitPosition for that adjacency reset when the BFR creates a copy for it. The BitPosition will still be reset for copies of the packet made towards other adjacencies. The can be used for example in ring topologies as explained below.

#### 3.2.2. Forward Routed

A "forward_routed" adjacency is an adjacency towards a BFR that is not a forward_connected adjacency: towards a loopback address of a BFR or towards an interface address that is non-directly connected. Forward_routed packets are forwarded via the Routing Underlay.

If the Routing Underlay has multiple paths for a forward_routed adjacency, it will perform ECMP independent of BIER-TE for packets forwarded across a forward_routed adjacency.

If the Routing Underlay has FRR, it will perform FRR independent of BIER-TE for packets forwarded across a forward_routed adjacency.

#### 3.2.3. ECMP

The ECMP mechanisms in BIER are tied to the BIER BIFT and are are therefore not directly useable with BIER-TE. The following procedures describe ECMP for BIER-TE that we consider to be lightweight but also well manageable. It leverages the existing entropy parameter in the BIER header to keep packets of the flows on the same path and it introduces a "seed" parameter to allow engineering traffic to be polarized or randomized across multiple hops.

An "Equal Cost Multipath" (ECMP) adjacency has a list of two or more adjacencies included in it. It copies the BIER-TE to one of those adjacencies based on the ECMP hash calculation. The BIER-TE ECMP hash algorithm must select the same adjacency from that list for all packets with the same "entropy" value in the BIER-TE header if the same number of adjacencies and same seed are given as parameters. Further use of the seed parameter is explained below.

#### 3.2.4. Local Decap

A "local_decap" adjacency passes a copy of the payload of the BIER-TE packet to the packets NextProto within the BFR (IPv4/IPv6, Ethernet,...). A local_decap adjacency turns the BFR into a BFER for matching packets. Local_decap adjacencies require the BFER to support routing or switching for NextProto to determine how to further process the packet.

### 3.3. Encapsulation considerations

Specifications for BIER-TE encapsulation are outside the scope of this document. This section gives explanations and guidelines.

Because a BFR needs to interpret the BitString of a BIER-TE packet differently from a BIER packet, it is necessary to distinguish BIER from BIER-TE packets. This is subject to definitions in BIER encapsulation specifications.

MPLS encapsulation [I-D.ietf-bier-mpls-encapsulation] for example assigns one label by which BFRs recognizes BIER packets for every (SI,subdomain) combination. If it is desirable that every subdomain can forward only BIER or BIER-TE packets, then the label allocation could stay the same, and only the forwarding model (BIER/BIER-TE) would have to be defined per subdomain. If it is desirable to support both BIER and BIER-TE forwarding in the same subdomain, then additional labels would need to be assigned for BIER-TE forwarding.

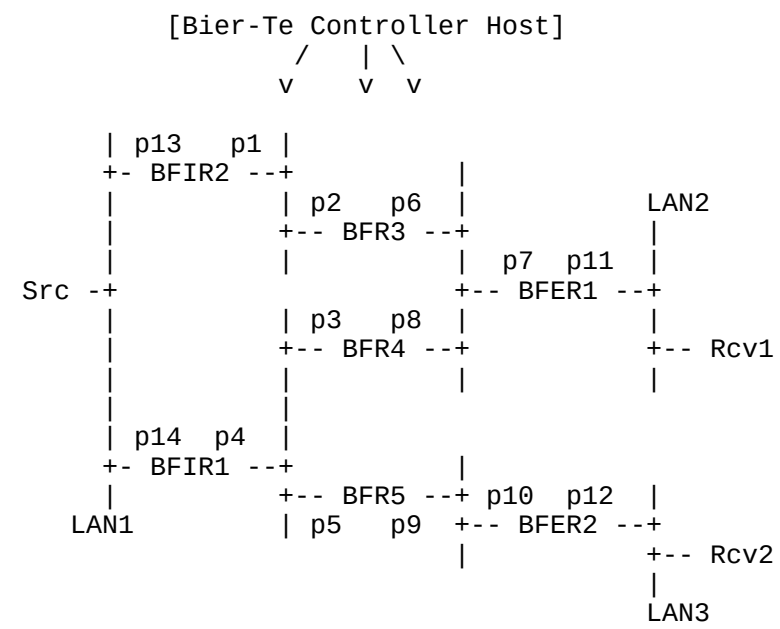
"forward_routed" requires an encapsulation permitting to unicast BIER-TE packets to a specific interface address on a target BFR. With MPLS encapsulation, this can simply be done via a label stack with that addresses label as the top label - followed by the label assigned to (SI,subdomain) - and if necessary (see above) BIER-TE. With non-MPLS encapsulation, some form of IP tunneling (IP in IP, LISP, GRE) would be required.

The encapsulation used for "forward_routed" adjacencies can equally support existing advanced adjacency information such as "loose source routes" via eg: MPLS label stacks or appropriate header extensions (eg: for IPv6).

### 3.4. Basic BIER-TE Forwarding Example

Step by step example of basic BIER-TE forwarding. This does not use ECMP or forward_routed adjacencies nor does it try to minimize the number of required BitPositions for the topology.

Picture 1: Forwarding Example




pXX indicate the BitPositions number assigned by the BIER-TE controller host to adjacencies in the BIER-TE topology. For example, p9 is the adjacency towards BFR9 on the LAN connecting to BFER2.

```
BIFT BFIR2:
  p13: local_decap()
   p2: forward_connected(BFR3)

BIFT BFR3:
   p1: forward_connected(BFIR2)
   p7: forward_connected(BFER1)
   p8: forward_connected(BFR4)

BIFT BFER1:
  p11: local_decap()
   p6: forward_connected(BFR3)
   p8: forward_connected(BFR4)
```

...and so on.

Traffic needs to flow from BFIR2 towards Rcv1, Rcv2. The controller determines it wants it to pass across the following paths:

```
                -> BFER1 ---------------> Rcv1
    BFIR2 -> BFR3
                -> BFR4 -> BFR5 -> BFER2 -> Rcv2
```

These paths equal to the following BitString: p2, p5, p7, p8, p10, p11, p12.

This BitString is set up in BFIR2. Multicast packets arriving at BFIR2 from Src are assigned this BitString.

BFIR2 forwards based on that BitString. It has p2 and p13 populated. Only p13 is in BitString which has an adjacency towards BFR3. BFIR2 resets p2 in BitString and sends a copy towards BFR2.

BFR3 sees a BitString of p5,p7,p8,p10,p11,p12. It is only interested in p1,p7,p8. It creates a copy of the packet to BFER1 (due to p7) and one to BFR4 (due to p8). It resets p7, p8 before sending.

BFER1 sees a BitString of p5,p10,p11,p12. It is only interested in p6,p7,p8,p11 and therefore considers only p11. p11 is a "local_decap" adjacency installed by the BIER-TE controller host because BFER1 should pass packets to IP multicast. The local_decap adjacency instructs BFER1 to create a copy, decapsulate it from the BIER header and pass it on to the NextProtocol, in this example IP multicast. IP multicast will then forward the packet out to LAN2 because it did receive PIM or IGMP joins on LAN2 for the traffic.

Further processing of the packet in BFR4, BFR5 and BFER2 accordingly.

## 4. BIER-TE Controller Host BitPosition Assignments

This section describes how the BIER-TE controller host can use the different BIER-TE adjacency types to define the BitPositions of a BIER-TE domain.

Because the size of the BitString is limiting the size of the BIER-TE domain, many of the options described exist to support larger topologies with fewer BitPositions (4.1, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8).

### 4.1. P2P Links

Each P2p link in the BIER-TE domain is assigned one unique BitPosition with a forward_connected adjacency pointing to the neighbor on the p2p link.

### 4.2. BFER

Every BFER is given a unique BitPosition with a local_decap adjacency.

### 4.3. Leaf BFERs

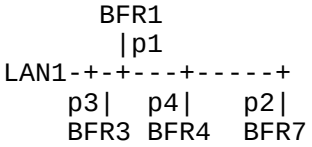
Leaf BFERs are BFERs where incoming BIER-TE packets never need to be forwarded to another BFR but are only sent to the BFER to exit the BIER-TE domain. For example, in networks where PEs are spokes connected to P routers, those PEs are Leaf BFIRs unless there is a U-turn between two PEs.

All leaf-BFER in a BIER-TE domain can share a single BitPosition. This is possible because the BitPosition for the adjacency to reach the BFER can be used to distinguish whether or not packets should reach the BFER.

This optimization will not work if an upstream interface of the BFER is using a BitPosition optimized as described in the following two sections (LAN, Hub and Spoke).

### 4.4. LANs

In a LAN, the adjacency to each neighboring BFR on the LAN is given a unique BitPosition. The adjacency of this BitPosition is a forward_connected adjacency towards the BFR and this BitPosition is populated into the BIFT of all the other BFRs on that LAN.




If Bandwidth on the LAN is not an issue and most BIER-TE traffic should be copied to all neighbors on a LAN, then BitPositions can be saved by assigning just a single BitPosition to the LAN and populating the BitPosition of the BIFTs of each BFRs on the LAN with a list of forward_connected adjacencies to all other neighbors on the LAN.

This optimization does not work in the face of BFRs redundantly connected to more than one LANs with this optimization because these BFRs would receive duplicates and forward those duplicates into the opposite LANs. Adjacencies of such BFRs into their LANs still need a separate BitPosition.

### 4.5. Hub and Spoke

In a setup with a hub and multiple spokes connected via separate p2p links to the hub, all p2p links can share the same BitPosition. The BitPosition on the hubs BIFT is set up with a list of forward_connected adjacencies, one for each Spoke.

This option is similar to the BitPosition optimization in LANs: Redundantly connected spokes need their own BitPositions.

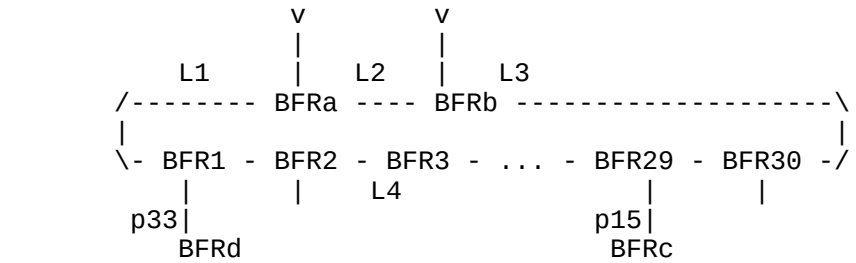
### 4.6. Rings

In L3 rings, instead of assigning a single BitPosition for every p2p link in the ring, it is possible to save BitPositions by setting the "Do Not Reset" (DNR) flag on forward_connected adjacencies.

For the rings shown in the following picture, a single BitPosition will suffice to forward traffic entering the ring at BFRa or BFRb all the way up to BFR1:

On BFRa, BFRb, BFR30,... BFR3, the BitPosition is populated with a forward_connected adjacency pointing to the clockwise neighbor on the ring and with DNR set. On BFR2, the adjacency also points to the clockwise neighbor BFR1, but without DNR set.

Handling DNR this way ensures that copies forwarded from any BFR in the ring to a BFR outside the ring will not have the ring BitPosition set, therefore minimizing the chance to create loops.



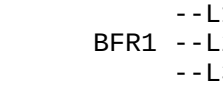

Note that this example only permits for packets to enter the ring at BFRa and BFRb, and that packets will always travel clockwise. If packets should be allowed to enter the ring at any ring BFR, then one would have to use two ring BitPositions. One for clockwise, one for counterclockwise.

Both would be set up to stop rotating on the same link, eg: L1. When the ingress ring BFR creates the clockwise copy, it will reset the counterclockwise BitPosition because the DNR bit only applies to the

bit for which the replication is done. Likewise for the clockwise BitPosition for the counterclockwise copy. In result, the ring ingress BFR will send a copy in both directions, serving BFRs on either side of the ring up to L1.

### 4.7. Equal Cost MultiPath (ECMP)

The ECMP adjacency allows to use just one BP per link bundle between two BFRs instead of one BP for each p2p member link of that link bundle. In the following picture, one BP is used across L1,L2,L3 and BFR1/BFR2 have for the BP



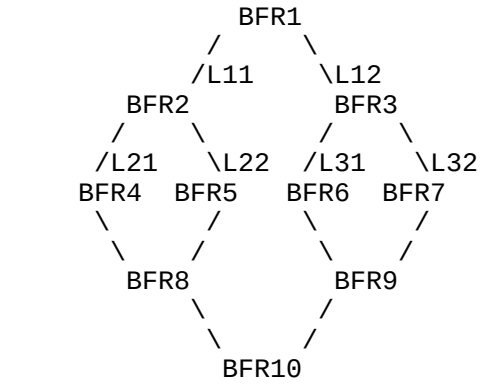

BIFT entry in BFR1:

| Index | Adjacencies |
|---|---|
| 0:6 | ECMP({L1-to-BFR2,L2-to-BFR2,L3-to-BFR2}, seed) |

BIFT entry in BFR2:

| Index | Adjacencies |
|---|---|
| 0:6 | ECMP({L1-to-BFR1,L2-to-BFR1,L3-to-BFR1}, seed) |

In the following example, all traffic from BFR1 towards BFR10 is intended to be ECMP load split equally across the topology. This example is not mean as a likely setup, but to illustrate that ECMP can be used to share BPs not only across link bundles, and it explains the use of the seed parameter.




BIFT entry in BFR1:

| | |
|---|---|
| 0:6 | ECMP({L11-to-BFR2,L12-to-BFR3}, seed) |

BIFT entry in BFR2:

| | |
|---|---|
| 0:6 | ECMP({L21-to-BFR4,L22-to-BFR5}, seed) |

BIFT entry in BFR3:

| | |
|---|---|
| 0:6 | ECMP({L31-to-BFR6,L32-to-BFR7}, seed) |

With the setup of ECMP in above topology, traffic would not be equally load-split. Instead, links L22 and L31 would see no traffic at all: BFR2 will only see traffic from BFR1 for which the ECMP hash in BFR1 selected the first adjacency in a list of 2 adjacencies: link L11-to-BFR2. When forwarding in BFR2 performs again an ECMP with two adjacencies on that subset of traffic, then it will again select the first of its two adjacencies to it: L21-to-BFR4. And therefore L22 and BFR5 sees no traffic.

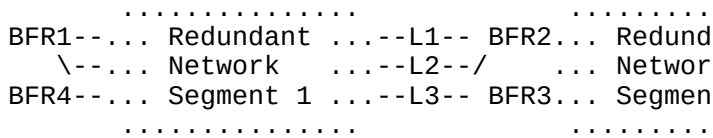
To resolve this issue, the ECMP adjacency on BFR1 simply needs to be set up with a different seed than the ECMP adjacencies on BFR2/BFR3

This issue is called polarization. It depends on the ECMP hash. It is possible to build ECMP that does not have polarization, for example by taking entropy from the actual adjacency members into account, but that can make it harder to achieve evenly balanced load-splitting on all BFR without making the ECMP hash algorithm potentially too complex for fast forwarding in the BFRs.

### 4.8. Routed adjacencies

#### 4.8.1. Reducing BitPositions

Routed adjacencies can reduce the number of BitPositions required when the traffic engineering requirement is not hop-by-hop explicit path selection, but loose-hop selection.




Assume the requirement in above network is to explicitly engineer paths such that specific traffic flows are passed from segment 1 to segment 2 via link L1 (or via L2 or via L3).

To achieve this, BFR1 and BFR4 are set up with a forward_routed adjacency BitPosition towards an address of BFR2 on link L1 (or link L2 BFR3 via L3).

For paths to be engineered through a specific node BFR2 (or BFR3), BFR1 and BFR4 are set up up with a forward_routed adjacency BitPosition towards a loopback address of BFR2 (or BFR3).

#### 4.8.2. Supporting nodes without BIER-TE

Routed adjacencies also enable incremental deployment of BIER-TE. Only the nodes through which BIER-TE traffic needs to be steered - with or without replication - need to support BIER-TE. Where they are not directly connected to each other, forward_routed adjacencies are used to pass over non BIER-TE enabled nodes.

## 5. Avoiding loops and duplicates

### 5.1. Loops

Whenever BIER-TE creates a copy of a packet, the BitString of that copy will have all BitPositions cleared that are associated with adjacencies in the BFR. This inhibits looping of packets. The only exception are adjacencies with DNR set.

With DNR set, looping can happen. Consider in the ring picture that link L4 from BFR3 is plugged into the L1 interface of BFRa. This creates a loop where the rings clockwise BitPosition is never reset for copies of the packets traveling clockwise around the ring.

To inhibit looping in the face of such physical misconfiguration, only forward_connected adjacencies are permitted to have DNR set, and the link layer destination address of the adjacency (eg.: MAC address) protects against closing the loop. Link layers without port unique link layer addresses should not used with the DNR flag set.

### 5.2. Duplicates

Duplicates happen when the topology of the BitString is not a tree but redundantly connecting BFRs with each other. The controller must therefore ensure to only create BitStrings that are trees in the topology.

When links are incorrectly physically re-connected before the controller updates BitStrings in BFIRs, duplicates can happen. Like loops, these can be inhibited by link layer addressing in forward_connected adjacencies.

If interface or loopback addresses used in forward_routed adjacencies are moved from one BFR to another, duplicates can equally happen. Such re-addressing operations must be coordinated with the controller.

## 6. BIER-TE FRR

FRR is an optional procedure. To leverage it, the BIER-TE controller host and BFRs need to support it. It does not have to be supported on all BFRs, but only those that are attached to a link/adjacency for which FRR support is required.

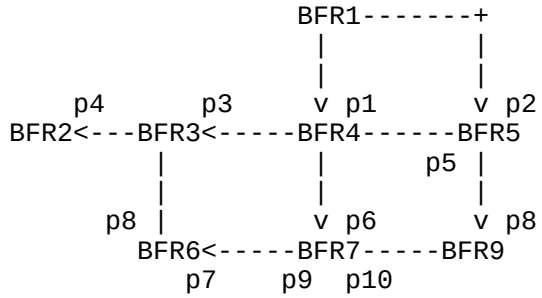
If BIER-TE FRR is supported by the BIER-TE controller host, then it needs to calculate the desired backup paths for link and/or node failures in the BIER-TE domain and download this information into the BIER-TE Adjacency FRR Table (BTAFT) of the BFRs. The BTAFT then drives FRR operations in the BIER-TE forwarding plane of that BFR.

The FRR operations modify the BIER header to facilitate local bypass of failed elements. In general, the backup is encoded in the bitstring of the packet. To avoid duplicates, it may be necessary to reset some bits in the bitstring or to use tunneling to the next-hops and next-next-hops of the multicast tree. Link and node failures can be addressed by the FRR mechanism.

Note that BIER-TE FRR does not require additional state depending on the multicast trees in the network but only depends on the network topology.

### 6.1. FRR Key Concepts

In this section we use the following example to explain the key concepts of BIER-TE FRR. The example shows a multicast tree from BFR1 to BFR2, BFR6, BFR9. The path to BFR2 is represented by the bits p1, p3 and p4. The bits p1, p7, p7 and the bits p2, p8 represent the path towards BFR6 and BFR 9, respectively. Local_decap bits for all BFR2,BFR6, and BFR9 are also used.




First, we consider that the link from P towards F fails. The failure can be protected by the backup paths over BFR3->BFR6->BFR7: p3, p8, p9 (BP1) and BFR5->BFR9->BFR7: p5, p8, p10 (BP2). The use of backup path BP1 does not cause duplicates. Backup path BP2 would cause duplicates because the local_decap bit for D2 is still set in bitstring at P. Two options exist to avoid duplicates. 1. We reset the local_decap bit for D2. This solution prevents the duplicate packet. However, this method can lead to lost packets in other examples. 2. We use a tunnel from P to F over D2 to prevent BIER packet processing at the nodes at the backup path. Tunnels can be implemented in two different ways.

1. A remote adjacency represented by a single bit which is a tunnel in the routing underlay. For an MPLS routing underlay, this can be implemented using an MPLS label stack. In the example we would introduce an additional bit (eg: p11) representing the tunnel.

2. BIER-in-BIER encapsulation using an additional BIER header with NextProto = BIER. BFRs need to support this feature. This methods does not require additional bits for remote adjacencies compared to remote adjacencies but it increases the size of the packet header. In this example the new bitstring contains the bits of BP2 and an additional local_decap bit for BFR7.

Now, we consider that BFR7 fails. The backup path must send the packets to all downstream next next-hops (DS-NNHs), i.e. the next-hops of the sub-tree rooted of BFR7. BFR4 can identify the DS-NNHs

by checking the bits of interest of the failed node BFR7. BFR6 is such a node because bit p7 is set. BFR9 is not downstream because there is no bit of interest from BFR7 to BFR9. Sending packets to BFR9 would causes duplicates because BFR9 is served using the branch BFR1->BFR5->BFR9.

Protection against link failures only requires knowledge of the failed adjacency. Protection against node failures requires additional knowledge of the downstream nodes of the tree. The computation of appropriate backup paths, AddBitmasks, ResetBitmasks, and BitPositions is outside of the scope of this document.

## 6.2. The BIER-TE Adjacency FRR Table (BTAFT)

The BIER-TE IF FRR Table exists in every BFR that is supporting BIER-TE FRR. It is indexed by FRR Adjacency Index that is compromised of the SI and the adjacency. Associated with each FRR Adjacency Index is the failed BitPosition (F-BP), Downstream BitPosition (DS-BP), ResetBitmask, and AddBitmask. The table can be configured to enable different actions for the AddBitMask. Either the table is configured to apply BIER-in-BIER encapsulation with a new BIER header containing the AddBitmask as new bitstring or to simply add the bits on the current bitstring.

| FRR Adjacency Index | Failed BP | Downstream BP | ResetBitmask | AddBitmask |
|---|---|---|---|---|
| 0:1 | 5 | 5 | ..0010000 | ..11000000 |

...

An FRR Adjacency is an adjacency that is used in the BIFT of the BFR. The BFR has to be able to determine whether the adjacency is up or down in less than 50msec. An FRR adjacency can be a forward_connected adjacency with fast L2 link state Up/Down state notifications or a forward_connected or forward_routed adjacency with a fast aliveness mechanism such as BFD. Details of those mechanism are outside the scope of this architecture.

The FRR Adjacency Index is the index that would be indicated on the fast Up/Down notifications to the BIER-TE forwarding plane and enables the selection of appropriate ResetBitMasks and AddBitmasks.

The failed BitPosition is the BP in the BIFT in which the FRR Adjacency is used. The downstream BitPosition is required to protect against node failures to identify the downstream adjacency as described in Section 6.1. The backup path/tree is constructed of the

individual ResetBitmasks and AddBitmasks of the downstream nodes. To protect against link failures, the DS-BP field is set equally to the F-BP field.

### 6.3. FRR in BIER-TE forwarding

The BIER-TE forwarding plane receives fast Up/Down notifications of BIER adjacencies which are used to with the FRR Adjacencies Index for different SIs. From the failed BitPosition in the BTAFT entry, it remembers which BPs are currently affected (have a down adjacency).

When a packet is received, BIER-TE forwarding checks if it has affected failed BPs and matching downstream BitPositions to which it would forward. If it does, it will remove the ResetBitmask bits from the packets BitString. Dependent on the table configuration it will either add the AddBitmask bits to the packets BitString or construct a new BIER header for rerouted packets. Note that the original packet must be still available for non-affected bitpositions.

Afterwards, normal BIER-TE forwarding occurs, taking the modified BitString or the additional BIER header into account. Note that the information is pre-computed by the controller and the BFR immediately bypasses a failure after its detection.

### 6.4. FRR in the BIER-TE Controller Host

The basic rules how the BIER-TE controller host would calculate ResetBitMask and AddBitmask are as follows:

1. The BIER-TE controller has to decide which tunnel mode a BFR uses for the BTAFT: remote adjacencies or BIER-in-BIER tunneling.

2. The BIER-TE controller host has to determine whether a failure of the adjacency should be taken to indicate link or node failure. This is a policy decision.

3. The ResetBitmask has the BitPosition of the failed adjacency.

4. In the case of link protection, the AddBitmask are the segments forming a path from the BFR over to the BFR on the other end of the failed link. The path can be formed using remote adjacencies for tunneling purposes.

5. In the case of node protection, the AddBitmask are the segments forming a tree from the BFR over to all necessary BFR downstream of the (assumed to be failed) BFR across the failed adjacency.

6. The ResetBitmask is extended with those segments that could lead to duplicate packets if the AddBitmask is added to possible BitStrings of packets using the failing BitPosition.

### 6.5. BIER-TE FRR Benefits

Compared to other FRR solutions, such as RSVP-TE/P2MP FRR, BIER-TE FRR has two key distinctions

- It maintains the goal of BIER-TE not to establish in-network per multicast traffic flow state. For that reason, the backup path/ trees are only tied to the topology but not to individual distribution trees.

- For the case of node failure, it allows to build a path engineered backup tree (4.) as opposed to only a set of p2p backup tunnels.

- BIER-in-BIER encapsulation enables backup tunnels in networks that do not provide a routing layer with tunneling capabilities. It may simplify network management because additional tunnels (such as GRE) must not be setup in the routing layer beforehand.

## 7. BIER-TE Forwarding Pseudocode

The following sections of Pseudocode are meant to illustrate the BIER-TE forwarding plane. This code is not meant to be normative but to serve both as a potentially easier to read and more precise representation of the forwarding functionality and to illustrate how simple BIER-TE forwarding is and that it can be efficiently be implemented.

The following procedure is executed on a BFR whenever the BIFT is changed by the BIER-TE controller host:

```
global MyBitsOfInterest

void BIFTChanged()
{
    for (Index = 0; Index++ ; Index <= BitStringLength)
        if(BIFT[Index] != <empty>)
            MyBitsOfInterest != 2<<(Index-1)
}
```

The following procedure is executed whenever an adjacency used for BIER-TE FRR changes state:

```
global ResetBitMaskByBT[BitStringLength]
global AddBitMaskByBT[BitStringLength]
global FRRaffectedBP

void FrrUpDown(FrrAdjacencyIndex, UpDown)
{
    global FRRAdjacenciesDown
    local Idx = FrrAdjacencyIndex

    if (UpDown == Up)
        FRRAdjacenciesDown &= ~ 2<<(FrrAdjacencyIndex-1)
    else
        FRRAdjacenciesDown |=   2<<(FrrAdjacencyIndex-1)

    for (Index = GetFirstBitPosition(FRRAdjacenciesDown); Index ;
        Index = GetNextBitPosition(FRRAdjacenciesDown, Index))

        local BP = BTAFT[Index].BitPosition
        FRRaffectedBP |= 2<<(Index)
        ResetBitMaskByBT[BP] |= BTAFT[Index].ResetBitMask
        AddBitMaskByBT[BP]   |= BTAFT[Index].AddBitMask
}
```

The following procedure is executed whenever a BIER-TE packet is to be forwarded:

```
void ForwardBierTePacket (Packet)
{
    // We calculate in BitMask the subset of BPs of the BitString
    // for which we have adjacencies. This is purely an
    // optimization to avoid to replicate for every BP
    // set in BitString only to discover that for most of them,
    // the BIFT has no adjacency.

    local BitMask = Packet->BitString
    Packet->BitString &= ~MyBitsOfInterest
    BitMask &= MyBitsOfInterest

    // FRR Operations
    // Note: this algorithm is not optimal yet for ECMP cases
    // it performs FRR replacement for all candidate ECMP paths

    local MyFRRBP = BitMask & FRRaffectedBP
    for (BP = GetFirstBitPosition(MyFRRNP); BP ;
         BP = GetNextBitPosition(MyFRRNP, BP))
        BitMask &= ~ResetBitMaskByBT[BP]
        BitMask |=  ResetBitMaskByBT[BP]

    // Replication
    for (Index = GetFirstBitPosition(BitMask); Index ;
         Index = GetNextBitPosition(BitMask, Index))
        foreach adjacency BIFT[Index]

            if(adjacency == ECMP(ListOfAdjacencies, seed) )
                I = ECMP_hash(sizeof(ListOfAdjacencies),
                              Packet->Entropy, seed)
                adjacency = ListOfAdjacencies[I]

            PacketCopy = Copy(Packet)

            switch(adjacency)
                case forward_connected(interface,neighbor,DNR):
                    if(DNR)
                        PacketCopy->BitString |= 2<<(Index-1)
                    SendToL2Unicast(PacketCopy,interface,neighbor)

                case forward_routed([VRF],neighbor):
                    SendToL3(PacketCopy,[VRF,]l3-neighbor)

                case local_decap([VRF],neighbor):
                    DecapBierHeader(PacketCopy)
                    PassTo(PacketCopy,[VRF,]Packet->NextProto)
}
```

# 8. Managing SI, subdomains and BFR-ids

When the number of bits required to represent the necessary hops in the topology and BFER exceeds the supported bitstring length, multiple SI and/or subdomains must be used. This section discusses how.

BIER-TE forwarding does not require the concept of BFR-id, but routing underlay, flow overlay and BIER headers may. This section also discusses how BFR-id can be assigned to BFIR/BFER for BIER-TE.

## 8.1. Why SI and sub-domains

For BIER and BIER-TE forwarding, the most important result of using multiple SI and/or subdomains is the same: Packets that need to be sent to BFER in different SI or subdomains require different BIER packets: each one with a bitstring for a different (SI,subdomain) bitstring. Each such bitstring uses one bitstring length sized SI block in the BIFT of the subdomain. We call this a BIFT:SI (block).

For BIER and BIER-TE forwarding itself there is also no difference whether different SI and/or sub-domains are chosen, but SI and subdomain have different purposes in the BIER architecture shared by BIER-TE. This impacts how operators are managing them and how especially flow overlays will likely use them.

By default, every possible BFIR/BFER in a BIER network would likely be given a BFR-id in subdomain 0 (unless there are > 64k BFIR/BFER).

If there are different flow services (or service instances) requiring replication to different subsets of BFER, then it will likely not be possible to achieve the best replication efficiency for all of these service instances via subdomain 0. Ideal replication efficiency for N BFER exists in a subdomain if they are split over not more than ceiling(N/bitstring-length) SI.

If service instances justify additional BIER:SI state in the network, additional subdomains will be used: BFIR/BFER are assigned BFIR-id in those subdomains and each service instance is configured to use the most appropriate subdomain. This results in improved replication efficiency for different services.

Even if creation of subdomains and assignment of BFR-id to BFIR/BFER in those subdomains is automated, it is not expected that individual service instances can deal with BFER in different subdomains. A service instance may only support configuration of a single subdomain it should rely on.

To be able to easily reuse (and modify as little as possible) existing BIER procedures including flow-overlay and routing underlay, when BIER-TE forwarding is added, we therefore reuse SI and subdomain logically in the same way as they are used in BIER: All necessary BFIR/BFER for a service use a single BIER-TE BIFT and are split across as many SI as necessary (see below). Different services may use different subdomains that primarily exist to provide more efficient replication (and for BIER-TE desirable traffic engineering) for different subsets of BFIR/BFER.

## 8.2. Bit assignment comparison BIER and BIER-TE

In BIER, bitstrings only need to carry bits for BFER, which lead to the model that BFR-ids map 1:1 to each bit in a bitstring.

In BIER-TE, bitstrings need to carry bits to indicate not only the receiving BFER but also the intermediate hops/links across which the packet must be sent. The maximum number of BFER that can be supported in a single bitstring or BIFT:SI depends on the number of bits necessary to represent the desired topology between them.

"Desired" topology because it depends on the physical topology, and on the desire of the operator to allow for explicit traffic engineering across every single hop (which requires more bits), or reducing the number of required bits by exploiting optimizations such as unicast (forward_route), ECMP or flood (DNR) over "uninteresting" sub-parts of the topology - eg: parts where different trees do not need to take different paths due to traffic-engineering reasons.

The total number of bits to describe the topology in a BIFT:SI can therefore easily be as low as 20% or as high as 80%. The higher the percentage, the higher the likelihood, that those topology bits are not just BIER-TE overhead without additional benefit, but instead they will allow to express the desired traffic-engineering alternatives.

## 8.3. Using BFR-id with BIER-TE

Because there is no 1:1 mapping between bits in the bitstring and BFER, BIER-TE can not simply rely on the BIER 1:1 mapping between bits in a bitstring and BFR-id.

In BIER, automatic schemes could assign all possible BFR-ids sequentially to BFERs. This will not work in BIER-TE. In BIER-TE, the operator or BIER-TE controller host has to determine a BFR-id for each BFER in each required subdomain. The BFR-id may or may not have a relationship with a bit in the bitstring. Suggestions are detailed below. Once determined, the BFR-id can then be configured on the

BFER and used by flow overlay, routing underlay and the BIER header almost the same as the BFR-id in BIER.

The one exception are application/flow-overlays that automatically calculate the bitstring(s) of BIER packets by converting BFR-id to bits. In BIER-TE, this operation can be done in two ways:

"Independent branches": For a given application or (set of) trees, the branches from a BFIR to every BFER are independent of the branches to any other BFER. For example, shortest part trees have independent branches.

"Interdependent branches": When a BFER is added or deleted from a particular distribution tree, branches to other BFER still in the tree may need to change. Steiner tree are examples of dependent branch trees.

If "independent branches" are sufficient, the BIER-TE controller host can provide to such applications for every BFR-id a SI:bitstring with the BIER-TE bits for the branch towards that BFER. The application can then independently calculate the SI:bitstring for all desired BFER by OR'ing their bitstrings.

If "interdependent branches" are required, the application could call a BIER-TE controller host API with the list of required BFER-id and get the required bitstring back. Whenever the set of BFER-id changes, this is repeated.

Note that in either case (unlike in BIER), the bits in BIER-TE may need to change upon link/node failure/recovery, network expansion and network load by other traffic (as part of traffic engineering goals). Interactions between such BFIR applications and the BIER-TE controller host do therefore need to support dynamic updates to the bitstrings.

### 8.4. Assigning BFR-ids for BIER-TE

For non-leaf BFER, there is usually a single bit k for that BFER with a local_decap() adjacency on the BFER. The BFR-id for such a BFER is therefore most easily the one it would have in BIER: SI * bitstring-length + k.

As explained earlier in the document, leaf BFER do not need such a separate bit because the fact alone that the BIER-TE packet is forwarded to the leaf BFER indicates that the BFER should decapsulate it. Such a BFER will have one or more bits for the links leading only to it. The BFR-id could therefore most easily be the BFR-id derived from the lowest bit for those links.

These two rules are only recommendations for the operator or BIER-TE controller assigning the BFR-ids. Any allocation scheme can be used, the BFR-ids just need to be unique across BFRs in each subdomain.

It is not currently determined if a single subdomain could or should be allowed to forward both BIER and BIER-TE packets. If this should be supported, there are two options:

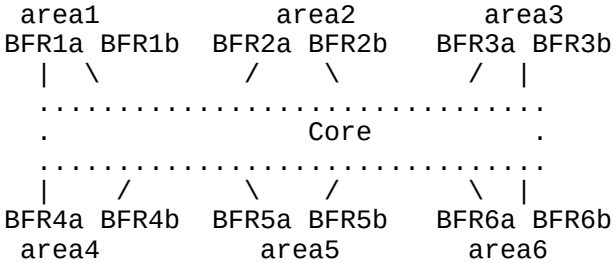
A. BIER and BIER-TE have different BFR-id in the same subdomain. This allows higher replication efficiency for BIER because their BFR-id can be assigned sequentially, while the bitstrings for BIER-TE will have also the additional bits for the topology. There is no relationship between a BFR BIER BFR-id and BIER-TE BFR-id.

B. BIER and BIER-TE share the same BFR-id. The BFR-id are assigned as explained above for BIER-TE and simply reused for BIER. The replication efficiency for BIER will be as low as that for BIER-TE in this approach. Depending on topology, only the same 20%..80% of bits as possible for BIER-TE can be used for BIER.

## 8.5. Example bit allocations

### 8.5.1. With BIER

Consider a network setup with a bitstring length of 256 for a network topology as shown in the picture below. The network has 6 areas, each with ca. 180 BFR, connecting via a core with some larger (core) BFR. To address all BFER with BIER, 4 SI are required. To send a BIER packet to all BFER in the network, 4 copies need to be sent by the BFIR. On the BFIR it does not make a difference how the BFR-id are allocated to BFER in the network, but for efficiency further down in the network it does make a difference.




With random allocation of BFR-id to BFER, each receiving area would (most likely) have to receive all 4 copies of the BIER packet because there would be BFR-id for each of the 4 SI in each of the areas. Only further towards each BFER would this duplication subside - when each of the 4 trees runs out of branches.

If BFR-id are allocated intelligently, then all the BFER in an area would be given BFR-id with as few as possible different SI. Each area would only have to forward one or two packets instead of 4.

Given how networks can grow over time, replication efficiency in an area will also easily go down over time when BFR-id are network wide allocated sequentially over time. An area that initially only has BFR-id in one SI might end up with many SI over a longer period of growth. Allocating SIs to areas with initially sufficiently many spare bits for growths can help to alleviate this issue. Or renumber BFR-id after network expansion. In this example one may consider to use 6 SI and assign one to each area.

This example shows that intelligent BFR-id allocation within at least subdomain 0 can even be helpful or even necessary in BIER.

#### 8.5.2. With BIER-TE

In BIER-TE one needs to determine a subset of the physical topology and attached BFER so that the "desired" representation of this topology and the BFER fit into a single bitstring. This process needs to be repeated until the whole topology is covered.

Once bits/SIs are assigned to topology and BFER, BFR-id is just a derived set of identifiers from the operator/BIER-TE controller as explained above.

Every time that different sub-topologies have overlap, bits need to be repeated across the bitstrings, increasing the overall amount of bits required across all bitstring/SIs. In the worst case, random subsets of BFER are assigned to different SI. This is much worse than in BIER because it not only reduces replication efficiency with the same number of overall bits, but even further - because more bits are required due to duplication of bits for topology across multiple SI. Intelligent BFER to SI assignment and selecting specific "desired" subtopologies can minimize this problem.

To set up BIER-TE efficiently for above topology, the following bit allocation methods can be used. This method can easily be expanded to other, similarly structured larger topologies.

Each area is allocated one or more SI depending on the number of future expected BFER and number of bits required for the topology in the area. In this example, 6 SI, one per area.

In addition, we use 4 bits in each SI: bia, bib, bea, beb: bit ingress a, bit ingress b, bit egress a, bit egress b. These bits will be used to pass BIER packets from any BFIR via any combination

of ingress area a/b BFR and egress area a/b BFR into a specific target area. These bits are then set up with the right forward_routed adjacencies on the BFIR and area edge BFR:

On all BFIR in an area j, bia in each BIFT:SI is populated with the same forward_routed(BFRja), and bib with forward_routed(BFRjb). On all area edge BFR, bea in BIFT:SI=k is populated with forward_routed(BFRka) and beb in BIFT:SI=k with forward_routed(BFRkb).

For BIER-TE forwarding of a packet to some subset of BFER across all areas, a BFIR would create at most 6 copies, with SI=1...SI=6, In each packet, the bits indicate bits for topology and BFER in that topology plus the four bits to indicate whether to pass this packet via the ingress area a or b border BFR and the egress area a or b border BFR, therefore allowing path engineering for those two "unicast" legs: 1) BFIR to ingress are edge and 2) core to egress area edge. Replication only happens inside the egress areas. For BFER in the same area as in the BFIR, these four bits are not used.

### 8.6. Summary

BIER-TE can like BIER support multiple SI within a sub-domain to allow re-using the concept of BFR-id and therefore minimize BIER-TE specific functions in underlay routing, flow overlay methods and BIER headers.

The number of BFIR/BFER possible in a subdomain is smaller than in BIER because BIER-TE uses additional bits for topology.

Subdomains can in BIER-TE be used like in BIER to create more efficient replication to known subsets of BFER.

Assigning bits for BFER intelligently into the right SI is more important in BIER-TE than in BIER because of replication efficiency and overall amount of bits required.

## 9. Further considerations

### 9.1. BIER-TE and existing FRR

BIER-TE as described above is an advanced method for node-protection where the replication in a failed node is on the fly replaced by another replication tree through bit operations on the BitString.

If BIER-TE FRR is not feasible or necessary, it is also possible for BIER-TE to leverage any existing form of "link" protection. For example: instead of directly setting up a forward_connected adjacency

to a next-hop neighbor, this can be a "protected" adjacency that is maintained by RSVP-TE (or another FRR mechanism) and passes via a backup path if the link fails.

BIER-in-BIER encapsulation provides P2MP protection in node failure cases because the new header can contain a new multicast. This allows for the least packet duplication if the routing underlay does not provide P2MP tunnels.

### 9.2. BIER-TE and Segment Routing

Segment Routing aims to achieve lightweight path engineering via loose source routing. Compared for example to RSVP-TE, it does not require per-path signaling to each of these hops.

BIER-TE is supports the same design philosophy for multicast. Like in SR, it relies on source-routing - via the definition of a BitString. Like SR, it only requires to consider the "hops" on which either replication has to happen, or across which the traffic should be steered (even without replication). Any other hops can be skipped via the use of routed adjacencies.

Instead of defining BitPositions for non-replicating hops, it is equally possible to use segment routing encapsulations (eg: MPLS label stacks) for "forward_routed" adjacencies.

## 10. Security Considerations

The security considerations are the same as for BIER with the following differences:

BFR-ids and BFR-prefixes are not used in BIER-TE, nor are procedures for their distribution, so these are not attack vectors against BIER-TE.

## 11. IANA Considerations

This document requests no action by IANA.

## 12. Acknowledgements

The authors would like to thank Greg Shepherd, Ijsbrand Wijnands and Neale Ranns for their extensive review and suggestions.

## 13. Change log [RFC Editor: Please remove]

03: Updated the FRR section. Added examples for FRR key concepts in Section 6.1. Added BIER-in-BIER tunneling as option for tunnels in backup paths. BIFT structure is expanded and contains an additional match field to support full node protection with BIER-TE FRR.

03: Updated Section 9.1. Explanation how BIER-in-BIER encapsulation provides P2MP protection for node failures even though the routing underlay does not provide P2MP.

02: Changed the definition of BIFT to be more inline with BIER. In revs. up to -01, the idea was that a BIFT has only entries for a single bitstring, and every SI and subdomain would be a separate BIFT. In BIER, each BIFT covers all SI. This is now also how we define it in BIER-TE.

02: Added Section 8 to explain the use of SI, subdomains and BFR-id in BIER-TE and to give an example how to efficiently assign bits for a large topology requiring multiple SI.

02: Added further detailed for rings - how to support input from all ring nodes.

01: Fixed BFIR -> BFER for section 4.3.

01: Added explanation of SI, difference to BIER ECMP, consideration for Segment Routing, unicast FRR, considerations for encapsulation, explanations of BIER-TE controller host and CLI.

00: Initial version.

## 14. References

[I-D.ietf-bier-architecture]
Wijnands, I., Rosen, E., Dolganow, A., P, T., and S. Aldrin, "Multicast using Bit Index Explicit Replication", draft-ietf-bier-architecture-03 (work in progress), January 2016.

[I-D.ietf-bier-mpls-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., and S. Aldrin, "Encapsulation for Bit Index Explicit Replication in MPLS Networks", draft-ietf-bier-mpls-encapsulation-03 (work in progress), February 2016.

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

## Authors' Addresses

Toerless Eckert
Cisco Systems, Inc.

Email: eckert@cisco.com

Gregory Cauchie
Bouygues Telecom

Email: GCAUCHIE@bouyguestelecom.fr

Wolfgang Braun
University of Tuebingen

Email: wolfgang.braun@uni-tuebingen.de

Michael Menth
University of Tuebingen

Email: menth@uni-tuebingen.de

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 20, 2016

L. Ginsberg, Ed.
Cisco Systems
A. Przygienda
Ericsson
S. Aldrin
Google
J. Zhang
Juniper Networks, Inc.

March 19, 2016

# BIER support via ISIS
# draft-ietf-bier-isis-extensions-02

## Abstract

Specification of an ISIS extension to support BIER domains and sub-domains.

## 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 RFC 2119 [RFC2119] .

## 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 20, 2016.

## Copyright Notice

Table of Contents



## 1. Introduction

Bit Index Explicit Replication (BIER) [I-D.draft-ietf-bier-architecture-03] defines an architecture where all intended multicast receivers are encoded as bitmask in the Multicast packet header within different encapsulations such as [I-D.draft-ietf-bier-mpls-encapsulation-03]. A router that receives such a packet will forward the packet based on the Bit Position in the packet header towards the receiver(s), following a precomputed tree for each of the bits in the packet. Each receiver is represented by a unique bit in the bitmask.

This document presents necessary extensions to the currently deployed ISIS for IP [RFC1195] protocol to support distribution of information necessary for operation of BIER domains and sub-domains. This document defines a new TLV to be advertised by every router participating in BIER signaling.

## 2. Terminology

Some of the terminology specified in [I-D.draft-ietf-bier-architecture-03] is replicated here and extended by necessary definitions:

BIER: Bit Index Explicit Replication (The overall architecture of forwarding multicast using a Bit Position).

BIER-OL: BIER Overlay Signaling. (The method for the BFIR to learn about BFER's).

BFR: Bit Forwarding Router (A router that participates in Bit Index Multipoint Forwarding). A BFR is identified by a unique BFR-prefix in a BIER domain.

BFIR: Bit Forwarding Ingress Router (The ingress border router that inserts the BM into the packet). Each BFIR must have a valid BFR-id assigned.

BFER: Bit Forwarding Egress Router. A router that participates in Bit Index Forwarding as leaf. Each BFER must be a BFR. Each BFER must have a valid BFR-id assigned.

BFT: Bit Forwarding Tree used to reach all BFERs in a domain.

BIFT: Bit Index Forwarding Table.

BMS: Bit Mask Set. Set containing bit positions of all BFER participating in a set.

BMP: Bit Mask Position, a given bit in a BMS.

Invalid BMP: Unassigned Bit Mask Position, consisting of all 0s.

IGP signalled BIER domain: A BIER underlay where the BIER synchronization information is carried in IGP. Observe that a multi-topology is NOT a separate BIER domain in IGP.

BIER sub-domain: A further distinction within a BIER domain identified by its unique sub-domain identifier. A BIER sub-domain can support multiple BitString Lengths.

BFR-id: An optional, unique identifier for a BFR within a BIER sub-domain.

Invalid BFR-id: Unassigned BFR-id, consisting of all 0s.

## 3. IANA Considerations

This document adds the following new sub-TLV to the registry of sub-TLVs for TLVs 235, 237 [RFC5120] and TLVs 135,236 [RFC5305],[RFC5308].

Value: 32 (suggested - to be assigned by IANA)

Name: BIER Info

This document also introduces a new registry for sub-sub-TLVs for the BIER Info sub-TLV added above. The registration policy is Expert Review as defined in [RFC5226]. This registry is part of the "IS-IS TLV Codepoints" registry. The name of the registry is "sub-sub-TLVs for BIER Info sub-TLV". The defined values are:

| Type | Name |
|---|---|
| 1 | BIER MPLS Encapsulation |
| 2 | BIER sub-domain Tree Type |
| 3 | BIER sub-domain BSL conversion |

## 4. Concepts

### 4.1. BIER Domains and Sub-Domains

An ISIS signalled BIER domain is aligned with the scope of distribution of BFR-prefixes that identify the BFRs within ISIS. ISIS acts in such a case as the supporting BIER underlay.

Within such a domain, the extensions defined in this document advertise BIER information for one or more BIER sub-domains. Each sub-domain is uniquely identified by a subdomain-id. Each subdomain is associated with a single ISIS topology [RFC5120], which may be any of the topologies supported by ISIS. Local configuration controls which <MT,SD> pairs are supported by a router. The mapping of sub-domains to topologies MUST be consistent within a BIER flooding domain.

Each BIER sub-domain has as its unique attributes the encapsulation used and the type of tree it is using to forward BIER frames (currently always SPF). Additionally, per supported bitstring length

in the sub-domain, each router will advertise the necessary label ranges to support it.

### 4.2. Advertising BIER Information

BIER information advertisements are associated with a new sub-TLV in the extended reachability TLVs. BIER information is always associated with a host prefix which MUST be a node address for the advertising node. The following restrictions apply:

- Prefix length MUST be 32 for an IPv4 prefix or 128 for an IPv6 prefix
- When the Prefix Attributes Flags sub-TLV is present N flag MUST be set and X and R flags MUST NOT be set. [RFC7794]
- BIER sub-TLVs MUST NOT be included when a prefix reachability advertisement is leaked between levels.

## 5. Procedures

### 5.1. Enabling a BIER Sub-Domain

A given sub-domain with identifier SD with supported bitstring lengths MLs in a multi-topology MT [RFC5120] is denoted further as <MT,SD,MLs> and does not have to be advertised by default by BFRs to preserve the scaling of the protocol (i.e. ISIS carries no TLVs containing any of the elements related to <MT,SD>). The advertisement may be triggered e.g. by a first BIER sub-TLV (Section 6.1) containing <MT,SD> advertised into the area. The specific trigger itself is outside the scope of this RFC but can be for example a VPN desiring to initiate a BIER sub-domain as MI-PMSI [RFC6513] tree or a pre-configured BFER (since BFERs will always advertise the BIER sub-TLV to make sure they can be reached). It is outside the scope of this document to describe what trigger for a router capable of participating in <MT,SD> is used to start the origination of the necessary information to join into it.

### 5.2. Multi Topology and Sub-Domain

A given sub-domain is supported within one and only one topology. All routers in the flooding scope of the BIER sub-TLVs MUST advertise the same sub-domain within the same multi-topology. A router receiving an <MT,SD> advertisement which does not match the locally configured pair MUST report a misconfiguration of the received <MT, SD> pair. All received BIER advertisements associated with the conflicting <MT, SD> pair MUST be ignored.

### 5.3. Encapsulation

All routers in the flooding scope of the BIER TLVs MUST advertise the same encapsulation for a given <MT,SD>. A router discovering encapsulation advertised that is different from its own MUST report a misconfiguration of a specific <MT,SD>. All received BIER advertisements associated with the conflicting <MT, SD> pair MUST be ignored.

### 5.4. Tree Type

All routers in the flooding scope of the BIER TLVs MAY advertise a supported tree type for a given <MT,SD>. Tree type indicates the algorithm used when calculating the optimal path. Currently only the default algorithm "SPF" is defined - which has a tree type of 0. If no tree type is advertised tree type 0 is assumed. The supported tree type MUST be consistent for all routers supporting a given <MT,SD>.

### 5.5. Label advertisements for MPLS Encapsulation

A router that desires to participate in <MT,SD> MUST advertise for each bitstring length it supports in <MT,SD> a label range size that guarantees to cover the maximum BFR-id injected into <MT,SD> (which implies a certain maximum set id per bitstring length as described in [I-D.draft-ietf-bier-architecture-03]). Any router that violates this condition MUST be excluded from BIER BFTs for <MT,SD>.

### 5.6. BFR-id Advertisements

Each BFER/BFIR MAY advertise with its TLV<MT,SD> the BFR-id that it has administratively chosen. A valid BFR-id MUST be unique within the flooding scope of the BIER advertisments. All BFERs/BFIRs MUST detect advertisement of duplicate valid BFR-IDs for a given <MT, SD>. When such duplication is detected all of the routers advertising duplicates MUST be treated as if they did not advertise a valid BFR-id. This implies they cannot act as BFER or BFIR in that <MT,SD>.

### 5.7. Reporting Misconfiguration

Whenever an advertisement is received which violates any of the constraints defined in this document the receiving router MUST report the misconfiguration.

### 5.8. Flooding Reduction

BIER domain information SHOULD change infrequently. Frequent changes will increase the number of Link State PDU (LSP) updates and negatively impact performance in the network.

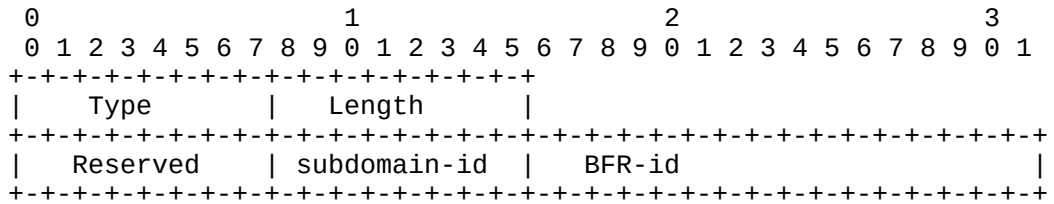
## 6. Packet Formats

All ISIS BIER information is carried within the TLVs 235, 237 [RFC5120] or TLVs 135 [RFC5305], or TLV 236 [RFC5308].

### 6.1. BIER Info sub-TLV

This sub-TLV carries the information for the BIER sub-domains that the router participates in as BFR. This sub-TLV MAY appear multiple times in a given prefix-reachability TLV - once for each sub-domain supported in the associated topology.

The sub-TLV advertises a single <MT,SD> combination followed by optional sub-sub-TLVs as described in the following sections.




Type: as indicated in IANA section.

Length: 1 octet.

Reserved: MUST be 0 on transmission, ignored on reception. May be used in future versions. 8 bits

subdomain-id: Unique value identifying the BIER sub-domain. 1 octet

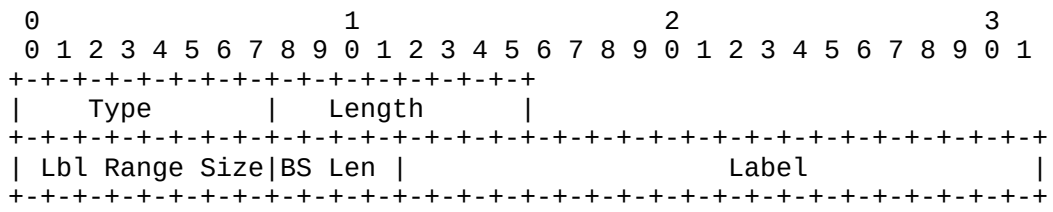
BFR-id: A 2 octet field encoding the BFR-id, as documented in [I-D.draft-ietf-bier-architecture-03]. If no BFR-id has been assigned this field is set to the invalid BFR-id.

### 6.2. BIER MPLS Encapsulation sub-sub-TLV

This sub-sub-TLV carries the information for the BIER MPLS encapsulation including the label range for a specific bitstring length for a certain <MT,SD>. It is advertised within the BIER Info sub-TLV (Section 6.1) . This sub-sub-TLV MAY appear multiple times within a single BIER info sub-TLV.

On violation of any of the following conditions, the receiving router MUST ignore the encapsulating BIER Info sub-TLV.

- Label ranges in multiple sub-sub-TLV MUST NOT overlap.
- Bitstring lengths in multiple sub-sub-TLVs MUST NOT be identical.
- The sub-sub-TLV MUST include the required bitstring lengths encoded in precisely the same way as in [I-D.draft-ietf-bier-mpls-encapsulation-03].
- The label range size MUST be greater than 0.
- All labels in the range MUST represent valid label values




Type: value of 1 indicating MPLS encapsulation.

Length: 1 octet.

Local BitString Length (BS Len): Encoded bitstring length as per [I-D.draft-ietf-bier-mpls-encapsulation-03]. 4 bits.

Label Range Size: Number of labels in the range used on encapsulation for this BIER sub-domain for this bitstring length, 1 octet.

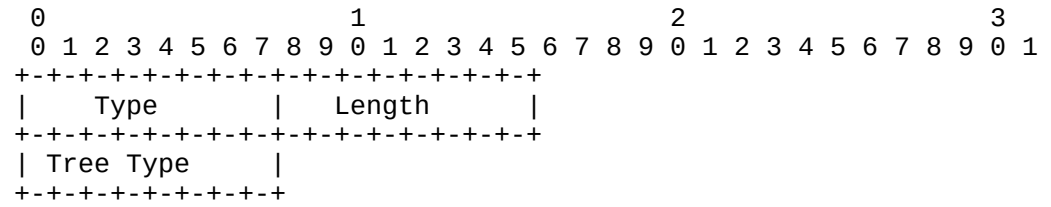
Label: First label of the range, 20 bits. The labels are as defined in [I-D.draft-ietf-bier-mpls-encapsulation-03].

### 6.3. Optional BIER sub-domain Tree Type sub-sub-TLV

This sub-sub-TLV carries the information associated with the supported BIER tree type for a <MT,SD> combination. It is carried within the BIER Info sub-TLV (Section 6.1) that the router participates in as BFR. This sub-sub-TLV is optional and its absence has the same semantics as its presence with Tree Type value 0 (SPF). When Tree Type 0 is used it is recommended that this sub-sub-TLV be omitted in order to reduce the space consumed in the parent TLV.

This sub-sub-TLV MUST NOT occur more than once in a BIER Info sub-TLV. If multiple occurences of this sub-sub-TLV are present in a single BIER Info sub-TLV the encapsulating BIER Info sub-TLV MUST be ignored.

If the tree type (implied or explicitly advertised) does not match the locally configured tree type associated with the matching <MT, SD> pair the encapsulating sub-TLV MUST be ignored.




Type: value of 1 indicating BIER Tree Type.

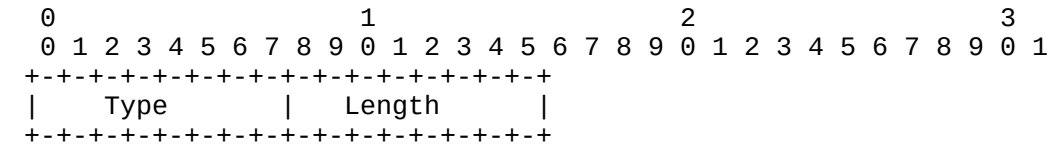
Length: 1 octet.

Tree Type: 1 octet

### 6.4. Optional BIER sub-domain BSL conversion sub-sub-TLV

This sub-sub-TLV indicates whether the BFR is capable of imposing a different Bit String Length (BSL) than the one it received in a BIER encapsulated packet. Such a capability may allow future, advanced tree types which ensure simple migration procedures from one BSL to another in a given <MT,SD> or prevent stable blackholes in scenarios where not all routers support the same set of BSLs in a given <MT,SD>. It is carried within the BIER Info sub-TLV (Section 6.1). This sub-sub-TLV is optional and its absence indicates that the router is NOT capable of imposing different BSLs but will always forward the packet with the BSL unchanged. This sub-sub-TLV MAY occur at most once in a given BIER info sub-TLV. If multiple

occurences of this sub-sub-TLV are received in a given BIER info sub-TLV the encapsulating sub-TLV MUST be ignored.




## 7. Security Considerations

Implementations must assure that malformed TLV and Sub-TLV permutations do not result in errors which cause hard protocol failures.

## 8. Acknowledgements

The RFC is aligned with the [I-D.draft-ietf-bier-ospf-bier-extensions-01] draft as far as the protocol mechanisms overlap.

Many thanks for comments from (in no particular order) Hannes Gredler, Ijsbrand Wijnands, Peter Psenak and Chris Bowers.

## 9. Normative References

[I-D.draft-ietf-bier-architecture-03]
Wijnands et al., IJ., "Stateless Multicast using Bit Index Explicit Replication Architecture", internet-draft draft-ietf-bier-architecture-03.txt, Jan 2016.

[I-D.draft-ietf-bier-mpls-encapsulation-03]
Wijnands et al., IJ., "Bit Index Explicit Replication using MPLS encapsulation", internet-draft draft-ietf-bier-mpls-encapsulation-03.txt, Feb 2016.

[I-D.draft-ietf-bier-ospf-bier-extensions-01]
Psenak et al., P., "OSPF Extension for Bit Index Explicit Replication", internet-draft draft-ietf-bier-ospf-bier-extensions-01.txt, October 2015.

[RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, DOI 10.17487/RFC1195, December 1990, <http://www.rfc-editor.org/info/rfc1195>.

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

[RFC5120] Przygienda, T., Shen, N., and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)", RFC 5120, DOI 10.17487/RFC5120, February 2008, <http://www.rfc-editor.org/info/rfc5120>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <http://www.rfc-editor.org/info/rfc5305>.

[RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, DOI 10.17487/RFC5308, October 2008, <http://www.rfc-editor.org/info/rfc5308>.

[RFC6513] Rosen, E., Ed. and R. Aggarwal, Ed., "Multicast in MPLS/ BGP IP VPNs", RFC 6513, DOI 10.17487/RFC6513, February 2012, <http://www.rfc-editor.org/info/rfc6513>.

[RFC7794] Ginsberg, L., Ed., Decraene, B., Previdi, S., Xu, X., and U. Chunduri, "IS-IS Prefix Attributes for Extended IPv4 and IPv6 Reachability", RFC 7794, DOI 10.17487/RFC7794, March 2016, <http://www.rfc-editor.org/info/rfc7794>.

## Authors' Addresses

Les Ginsberg (editor)
Cisco Systems
510 McCarthy Blvd.
Milpitas, CA 95035
USA

Email: ginsberg@cisco.com

Tony Przygienda
Ericsson
300 Holger Way
San Jose, CA 95134
USA

Email: antoni.przygienda@ericsson.com

Sam Aldrin
Google
1600 Amphitheatre Parkway
Mountain View, CA
USA

Email: aldrin.ietf@gmail.com

Jeffrey (Zhaohui) Zhang
Juniper Networks, Inc.
10 Technology Park Drive
Westford, MA 01886
USA

Email: zzhang@juniper.net

OSPF
Internet-Draft
Intended status: Standards Track
Expires: September 22, 2016

P. Psenak, Ed.
N. Kumar
IJ. Wijnands
Cisco
A. Dolganow
Alcatel-Lucent
T. Przygienda
Ericsson
J. Zhang
Juniper Networks, Inc.
S. Aldrin
Google, Inc.
March 21, 2016

# OSPF Extensions for BIER
draft-ietf-bier-ospf-bier-extensions-02.txt

## Abstract

Bit Index Explicit Replication (BIER) is an architecture that provides multicast forwarding through a "BIER domain" without requiring intermediate routers to maintain multicast related per-flow state. Neither does BIER require an explicit tree-building protocol for its operation. A multicast data packet enters a BIER domain at a "Bit-Forwarding Ingress Router" (BFIR), and leaves the BIER domain at one or more "Bit-Forwarding Egress Routers" (BFERs). The BFIR router adds a BIER header to the packet. Such header contains a bit-string in which each bit represents exactly one BFER to forward the packet to. The set of BFERs to which the multicast packet needs to be forwarded is expressed by the according set of bits switched on in BIER packet header.

This document describes the OSPF protocol extension required for BIER with MPLS encapsulation.

## 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 22, 2016.



Table of Contents



## 1. Introduction

Bit Index Explicit Replication (BIER) is an architecture that provides optimal multicast forwarding through a "BIER domain" without requiring intermediate routers to maintain any multicast related per-flow state. Neither does BIER explicitly require a tree-building protocol for its operation. A multicast data packet enters a BIER domain at a "Bit-Forwarding Ingress Router" (BFIR), and leaves the BIER domain at one or more "Bit-Forwarding Egress Routers" (BFERs). The BFIR router adds a BIER header to the packet. The BIER header contains a bit-string in which each bit represents exactly one BFER to forward the packet to. The set of BFERs to which the multicast packet needs to be forwarded is expressed by setting the bits that correspond to those routers in the BIER header.

BIER architecture requires routers participating in BIER to exchange BIER related information within a given domain. BIER architecture permits link-state routing protocols to perform distribution of such information. This document describes extensions to OSPF necessary to carry BIER specific information in the case where BIER uses MPLS encapsulation as described in [I-D.wijnands-mpls-bier-encapsulation].

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. Flooding of the BIER Information in OSPF

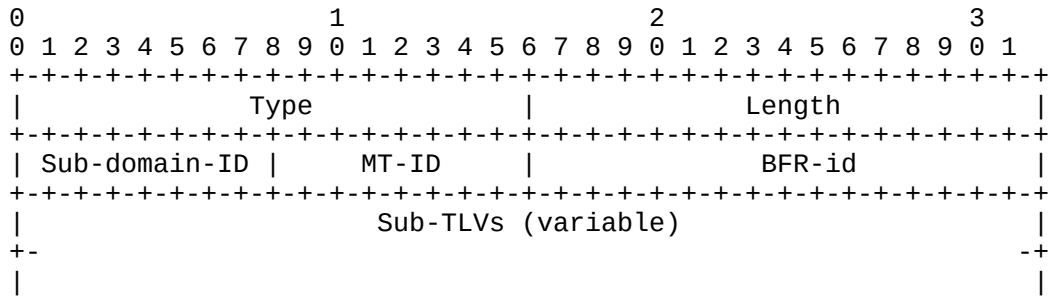
All BIER specific information that a BFR needs to advertise to other BFRs is associated with a BFR-Prefix. BFR prefix is a unique (within a given BIER domain), routable IP address that is assigned to each BFR as described in more detail in section 2 of [I-D.wijnands-bier-architecture].

Given that BIER information must be associated with a BFR prefix, the OSPF Extended Prefix Opaque LSA [I-D.ietf-ospf-prefix-link-attr] has been chosen to flood it.

### 2.1. The BIER Sub-TLV

A new Sub-TLV of the Extended Prefix TLV (defined in [I-D.ietf-ospf-prefix-link-attr]) is defined for distributing BIER information. The new Sub-TLV is called BIER Sub-TLV. Multiple BIER Sub-TLVs may be included in the Extended Prefix TLV.

BIER Sub-TLV has the following format:




Type: TBD

Length: variable

Sub-domain-ID: Unique value identifying the BIER sub-domain within the BIER domain, as described in section 1 of [I-D.wijnands-bier-architecture].

MT-ID: Multi-Topology ID (as defined in [RFC4915]) that identifies the topology that is associated with the BIER sub-domain.

BFR-id: A 2 octet field encoding the BFR-id, as documented in section 2 [I-D.wijnands-bier-architecture]. If the BFR is not locally configured with a valid BFR-id, the value of this field is set to invalid BFR-id per [I-D.wijnands-bier-architecture].

Each BFR sub-domain MUST be associated with one and only one OSPF topology that is identified by the MT-ID. If the association between BIER sub-domain and OSPF topology advertised in the BIER sub-TLV by other BFRs is in conflict with the association locally configured on the receiving router, whole BIER sub-TLV of the advertising routers MUST be ignored.

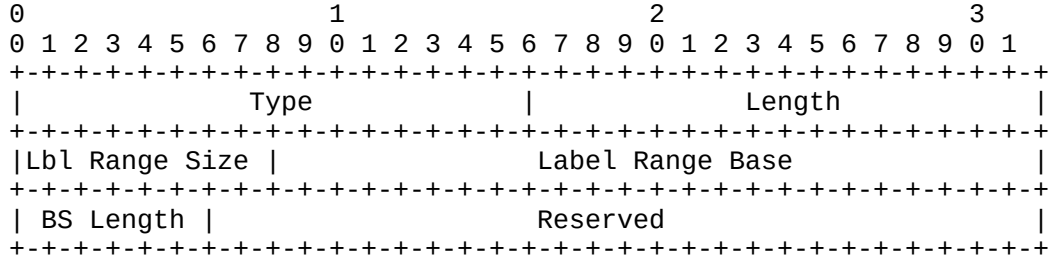
If a BFR advertises the same Sub-domain-ID in multiple BIER sub-TLVs, the BRF MUST be treated as if it did not advertise a BIER sub-TLV for such sub-domain.

All BFRs MUST detect advertisement of duplicate valid BFR-IDs for a given MT-ID and Sub-domain-ID. When such duplication is detected all BFRs advertising duplicates MUST be treated as if they did not advertise a valid BFR-id.

## 2.2. The BIER MPLS Encapsulation Sub-TLV

BIER MPLS Encapsulation Sub-TLV is a sub-TLV of the BIER Sub-TLV. BIER MPLS Encapsulation Sub-TLV is used in order to advertise MPLS specific information used for BIER. It MAY appear multiple times in the BIER Sub-TLV.

BIER MPLS Encapsulation Sub-TLV has the following format:




Type: TBD

Length: 4 bytes

Label Range Size: A 1 octet field encoding the label range size of the label range. It MUST be greater then 0, otherwise the advertising router MUST be treated as if it did not advertise a BIER sub-TLV.

Label Range Base: A 3 octet field, where the 20 rightmost bits represent the first label in the label range.

BS Length: A 1 octet field encoding the supported BitString length associated with this BFR-prefix. The values allowed in this field are specified in section 3 of [I-D.wijnands-mpls-bier-encapsulation].

The "label range" is the set of labels beginning with the label range base and ending with (label range base)+(label range size)-1. A unique label range is allocated for each BitStream length and Sub-domain-ID. These labels are used for BIER forwarding as described in [I-D.wijnands-bier-architecture] and [I-D.wijnands-mpls-bier-encapsulation].

The size of the label range is determined by the number of Set Identifiers (SI) (section 2 of [I-D.wijnands-bier-architecture]) that are used in the network. Each SI maps to a single label in the label range. The first label is for SI=0, the second label is for SI=1, etc.

If same BS length is repeated in multiple BIER MPLS Encapsulation Sub-TLV inside the same BIER Sub-TLV, the advertising router MUST be treated as if it did not advertise a BIER sub-TLV.

Label ranges within all BIER MPLS Encapsulation Sub-TLV inside the same BIER Sub-TLV MUST NOT overlap. If the overlap is detected, the advertising router MUST be treated as if it did not advertise a BIER sub-TLV.

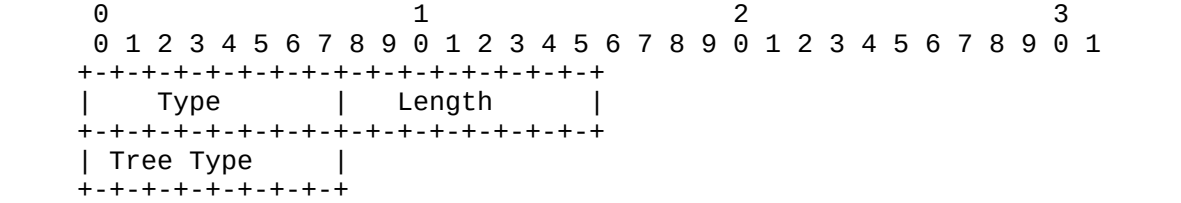
All advertised labels MUST be valid, otherwise the advertising router MUST be treated as if it did not advertise a BIER sub-TLV.

## 2.3. Optional BIER Tree Type Sub-TLV

This Sub-TLV carries the information associated with the supported BIER tree type for a subdomain. This Sub-TLV is optional and its absence has the same semantics as its presence with Tree Type value 0 (SPF). When Tree Type 0 is used it is recommended that this Sub-TLV is omitted in order to reduce the space consumed in the parent TLV.

This Sub-TLV MAY occur no more than once in a BIER sub-TLV. If multiple occurences of this Sub-TLV are present in a single BIER Sub-TLV the advertising router MUST be treated as if it did not advertise a BIER sub-TLV.

If the tree type (implied or explicitly advertised) does not match the locally configured tree type associated with the matching subdomain the advertising router MUST be treated as if it did not advertise a BIER sub-TLV.




Type: value of 1 indicating BIER Tree Type.

Length: 1 octet.

Tree Type: 1 octet

## 2.4. Flooding scope of BIER Information

Flooding scope of the OSPF Extended Prefix Opaque LSA [I-D.ietf-ospf-prefix-link-attr] that is used for advertising BIER Sub TLV is set to area. To allow BIER deployment in a multi-area environment, OSPF must propagate BIER information between areas. The following procedure is used in order to propagate BIER related information between areas:

When an OSPF ABR advertises a Type-3 Summary LSA from an intra-area or inter-area prefix to all its connected areas, it will also originate an Extended Prefix Opaque LSA, as described in [I-D.ietf-ospf-prefix-link-attr]. The flooding scope of the Extended Prefix Opaque LSA type will be set to area-scope. The route-type in the OSPF Extended Prefix TLV is set to inter-area. When determining whether a BIER Sub-TLV should be included in this LSA ABR will:

- look at its best path to the prefix in the source area and find the advertising router associated with the best path to that prefix.

- determine if such advertising router advertised a BIER Sub-TLV for the prefix. If yes, ABR will copy the information from

such BIER MPLS Sub-TLV when advertising BIER MPLS Sub-TLV to each connected area.

## 3. Security Considerations

Implementations must assure that malformed TLV and Sub-TLV permutations do not result in errors which cause hard OSPF failures.

## 4. IANA Considerations

The document requests three new allocations from the OSPF Extended Prefix sub-TLV registry as defined in [I-D.ietf-ospf-prefix-link-attr].

BIER Sub-TLV: TBD

BIER MPLS Encapsulation Sub-TLV: TBD

BIER Tree Type Sub-TLV: TBD

## 5. Acknowledgments

The authors would like to thank Rajiv Asati, Christian Martin, Greg Shepherd and Eric Rosen for their contribution.

## 6. Normative References

[I-D.ietf-ospf-prefix-link-attr]
Psenak, P., Gredler, H., rjs@rob.sh, r., Henderickx, W., Tantsura, J., and A. Lindem, "OSPFv2 Prefix/Link Attribute Advertisement", draft-ietf-ospf-prefix-link-attr-13 (work in progress), August 2015.

[I-D.wijnands-bier-architecture]
Wijnands, I., Rosen, E., Dolganow, A., and T. Przygienda, "Multicast using Bit Index Explicit Replication", draft-wijnands-bier-architecture-00 (work in progress), September 2014.

[I-D.wijnands-mpls-bier-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., and J. Tantsura, "Encapsulation for Bit Index Explicit Replication in MPLS Networks", draft-wijnands-mpls-bier-encapsulation-00 (work in progress), September 2014.

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

[RFC4915] Psenak, P., Mirtorabi, S., Roy, A., Nguyen, L., and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF", RFC 4915, DOI 10.17487/RFC4915, June 2007, <http://www.rfc-editor.org/info/rfc4915>.

## Authors' Addresses

Peter Psenak (editor)
Cisco
Apollo Business Center
Mlynske nivy 43
Bratislava 821 09
Slovakia

Email: ppsenak@cisco.com

Nagendra Kumar
Cisco
7200 Kit Creek Road
Research Triangle Park, NC 27709
US

Email: naikumar@cisco.com

IJsbrand Wijnands
Cisco
De Kleetlaan 6a
Diegem 1831
Belgium

Email: ice@cisco.com

Andrew Dolganow
Alcatel-Lucent
600 March Rd.
Ottawa, Ontario K2K 2E6
Canada

Email: andrew.dolganow@alcatel-lucent.com

Tony Przygienda
Ericsson
300 Holger Way
San Jose, CA 95134
USA

Email: antoni.przygienda@ericsson.com

Jeffrey Zhang
Juniper Networks, Inc.
10 Technology Park Drive
Westford, MA 01886
USA

Email: zzhang@juniper.net

Sam Aldrin
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA
USA

Email: aldrin.ietf@gmail.com

BIER WG
Internet-Draft
Intended status: Standards Track
Expires: September 16, 2016

C. Wang
Z. Zhang
ZTE Corporation
A. Qu

March 15, 2016

# BIER Ethernet
## draft-wang-bier-ethernet-01

Abstract

Bit Index Explicit Replication (BIER) [I-D.ietf-bier-architecture] is an architecture that provides optimal multicast forwarding through a "BIER domain" without requiring intermediate routers to maintain any multicast related per-flow state. BIER also does not require any explicit tree-building protocol for its operation. When a multicast data packet enters the BIER domain, the BFIR determines the BFERs to which the packet needs to be sent. Then the BFIR encapsulates the packet in a BIER header and forwards the packet according to the BIFTs. Currently, there is a BIER-MPLS solution to transmit multicast traffic using MPLS label indication. Alternatively, this document tries to propose a solution named BIER Ethernet to support BIER forwarding in Ethernet network.

Table of Contents

## 1. Introduction

Bit Index Explicit Replication (BIER) [I-D.ietf-bier-architecture] is an architecture that provides optimal multicast forwarding through a "BIER domain" without requiring intermediate routers to maintain any multicast related per-flow state. BIER also does not require any explicit tree-building protocol for its operation. A multicast data packet enters a BIER domain at a "Bit-Forwarding Ingress Router" (BFIR), and leaves the BIER domain at one or more "Bit-Forwarding Egress Routers" (BFERs). The BFIR encapsulates a BIER header to the packet. The BIER header contains a BitString in which each bit represents exactly one BFER to forward the packet to.

Specifically, after encapsulating a BIER header to the original multicast data packet, the BFIR and the intermediate BFRs as well as the BFERs require to carry out the BIER forwarding procedures to the BIER-encapsulated packet according to the information in the BIER header. As described in [I-D.ietf-bier-architecture], each BFR firstly should determine the packet’s Subdomain-ID, BitStringLength and Set ID information to locate the exact "Bit Index Forwading Table" (BIFT), and then do the subsequent procedures in terms of BitString and the found BIFT.

However the existing draft requires MPLS label preceding the BIER header and using Bottom label to carry BIER forwarding information is not clean cut deign.

Hence we should design BIER header that holds all BIER forwarding related information, and just let MPLS as an independent layer protocol to help BIER forwarding as it does for IPv4/IPv6/IPmcast traffic.

Additionally, the BIER forwarding capability will be also introduced in enterprise/data center, such feature may be newly implemented in switch ASICs, with clean cut design using BIER-ethernet draft, the implementation will be more clean as well.

So this document tries to propose this kind of BIER header which contains significant BIER information directly such as Subdomain-ID, BitStringLength and Set ID as well as BitString. It is applicable when a given BIER domain is an Ethernet network.

## 2. Convention and Terminology

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

The terms about BIER are defined in [I-D.ietf-bier-architecture].

## 3. BIER Ethernet Header

The BIER Ethernet header is shown in Figure 1.

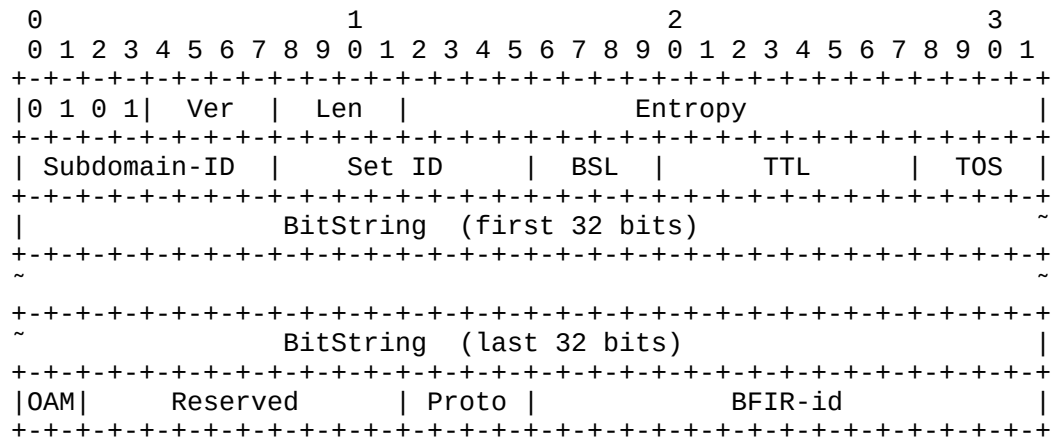



Figure 1: BIER Ethernet Header

First nibble: The first 4 bits of the header are set to 0101; this ensures that the BIER header will not be confused with an IP header or with the header of a pseudowire packet.

Ver: this 4-bit field identifies the version of the BIER header.

Len: This 4-bit field encodes the length of BIER Ethenet header.

Entropy: This 20-bit field specifies an "entropy" value that can be used for load balancing purposes. The BIER forwarding process may do equal cost load balancing, but the load balancing procedure MUST choose the same path for any two packets have the same entropy value.

Subdomain-ID: unique value identifying the BIER Subdomain within the BIER domain, as described in section 1 of [I-D.ietf-bier-architecture].

Set ID: indicates the packet’s Set Identifier.

BSL: indicates the packet’s BitStringLength.

TTL: Time to Live

TOS: Type of Service. It can be used to differentiate services to

different BIER packets.

BitString: together with the packet's Set ID, identifies the destination BFERs for this packet.

OAM: These two bits are used for the passive performance measurement marking method.

Reserved: These 10 bits are currently unused. They SHOULD be set to zero upon transmission, and MUST be ignored upon reception.

Proto:This 4-bit field identifies the type of the payload. (The "payload" is the packet or frame immediately following the BIER header.)

BFIR-id: By default, this is the BFR-id of the BFIR, in the Subdomain to which the packet has been assigned. The BFR-id is encoded in the 16-bit field as an unsigned integer in the range [1,65535].

Furthermore, BIER Ethernet encapsulated packet has the following format. The original multicast data packet is encapsulated with two headers (starting from the outermost header): Outer Ethernet Header + BIER Ethernet Header. Figure 2 is an example of an outer Ethernet Header. The outer VLAN tag is optional. In some situations, there may be some other encapsulation headers before the multicast data packet and after the BIER Ethernet header.

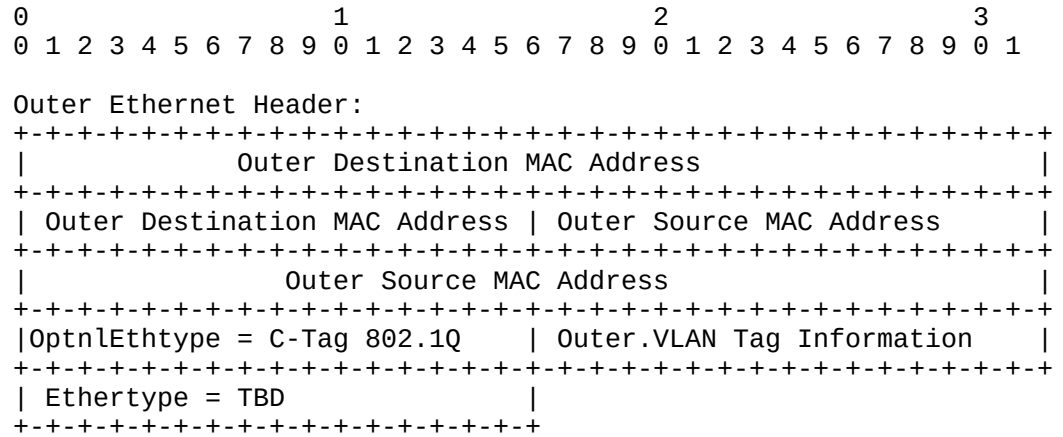



Figure 2: Outer Ethernet Header

Ethertype: requires a new Ethertype for BIER Ethernet header.

## 4. Imposing and processing the BIER Ethernet header

When a BFIR receives a multicast packet from outside the BIER domain, the BFIR carries out the following procedure:

1.By consulting the "multicast flow overlay", the BFIR determines the set of BFERs that must receive the packet.

2.By consulting the "BIER layer", the BFIR determines the packet's Subdomain, BitStringLength, Set Identifier and BitString information. The BFERs have the same Set Identifier can be encoded into the same BitString.

3.Using information provided by the routing underlay associated with the packet's BIER information, the BFIR determines the next hop for each (Set Identifier, the BitString) combination, and copies packet to each Set Identifier.

4.Before transmitting the packet to the next hop, the BFIR updates the BitString information and encapsulates the BIER Ethernet header to the multicast packet.

When an intermediate BFR receives a BIER Ethernet encapsulated packet, it acquires Subdomain-ID, BitStringLength as well as Set Identifier information directly from the BIER Ethernet header to determine the BIFT, and then forwards the received BIER packet according to the procedures described in [I-D.ietf-bier-architecture].

When a BFR receives a BIER Ethernet encapsulated packet whose Subdomain ID, Set Identifier and BitString identify the BFR itself, then the BFR is also a BFER for that packet. As a BFER, it must decapsulate the BIER Ethernet header, and pass the original multicast packet out.

## 5. Control Plane Considerations about BIER Ethernet

As described in the BIER OSPF extensions [I-D.ietf-bier-ospf-bier-extensions] and BIER ISIS extensions [I-D.ietf-bier-isis-extensions], they already define BIER Info Sub-TLV as the following format in Figure 3(take the ospf extensions for example).

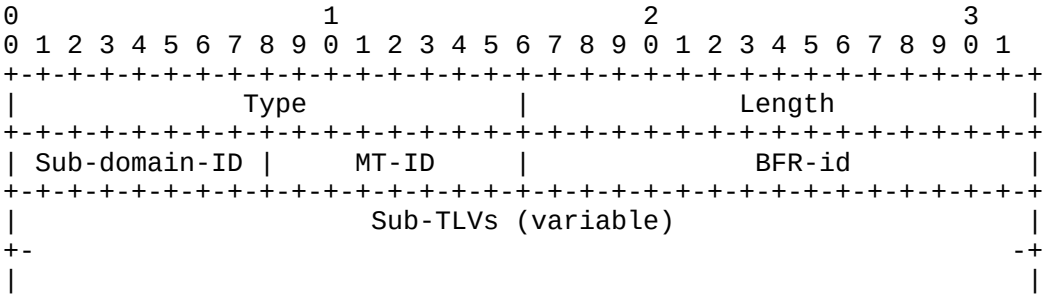



Figure 3: BIER Info Sub-TLV

To implement BIER Ethernet, the control plane, except the advertisements of BIER Info Sub-TLV, should have advertisements about BitStringLengths information the sending BFR supports. A reference format of BSL Sub-sub-TLV is illustrated in Figure 4 (take the ospf extensions for example as well).

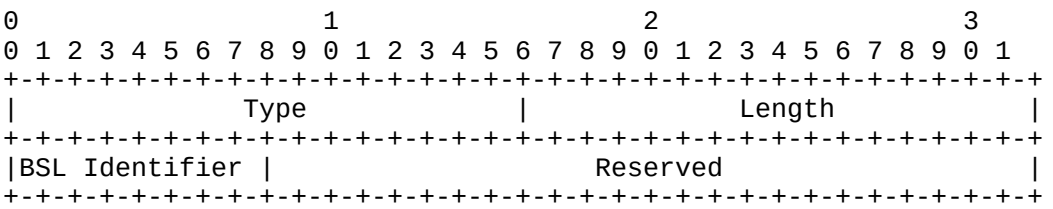



Figure 4: BSL Sub-sub-TLV

Type: value of 1 indicating BSL Sub-sub-TLV for BIER Ethernet encapsulation.

Len: This 8-bit field encodes the length of this sub-sub-TLV.

BSL Identifier: indicating the BSL the sending BFR supporting.

The sending BFR may support one or several BSLs, as following:

00000001: represents BSL 64 bits;

00000010: represents BSL 128 bits;

00000100: represents BSL 256 bits;

00001000: represents BSL 512 bits;

00010000: represents BSL 1024 bits;

00100000: represents BSL 2048 bits;

01000000: represents BSL 4096 bits;

Each bit represents one BSL. When there are two or more bits set, that means the sending BFR supports more than one BSL. For example, if the BSL Identifier is 00010101, it means the sending BFR supports 1024 bits, 256 bits and 64 bits.

Additionally, the similar BSL Sub-sub-TLV extension can be introduced in ISIS extension and IDR extension for implementing BIER Ethernet.

# 6. BIER Ethernet Considerations

## 6.1. BIER Ethernet for Traffic Enginerring

Specifically, BIER-TE encapsulation format may be the same as BIER encapsulation. However,how to interpret the BitString is totally different. Hence, BIER-Ethernet encapsulation MUST need one identifier to be assigned to identify the BIER header is for BIER forwarding or BIER-TE forwarding. For example, one bit in Reserved field can be reserved for this purpose.

## 6.2. BIER Ethernet for Multicast VPN

In MVPN, the P-tunnels are used for carrying multicast traffic across backbone. BIER tunnel Type is newly defined in [I-D.ietf-bier-mvpn]. The BIER Encapsulation used for multicast tunnel is independant of the (upstream assigned) MPLS label. Hence, BIER-Ethernet can also be used as P-Tunnel. In other words, there may need a new Tunnel Type to identify BIER-Ethernet Tunnel type, or a new flag to distinguish BIER-MPLS tunnel and BIER-Ethernet Tunnel.

# 7. Assignment Considerations

## 7.1. IEEE Registration Authority Considerations

This document requests the IEEE Registration Authority to assign a new Ethertype for BIER Ethernet Header.

## 7.2. IANA Considerations

This document requires new IANA allocation for BSL Sub-sub-TLV extension in different routing protocol.

## 8. Acknowledgements

The authors would like to thank IJsbrand Wijnands,Tony Przygienda and Andrew Qu for their ideas and contribution to this document.

# 9. References

## 9.1. Normative References

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

[RFC4915] Psenak, P., Mirtorabi, S., Roy, A., Nguyen, L., and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF", RFC 4915, DOI 10.17487/RFC4915, June 2007, <http://www.rfc-editor.org/info/rfc4915>.

## 9.2. Informative References

[I-D.ietf-bier-architecture] Wijnands, I., Rosen, E., Dolganow, A., P, T., and S. Aldrin, "Multicast using Bit Index Explicit Replication", draft-ietf-bier-architecture-03 (work in progress), January 2016.

[I-D.ietf-bier-isis-extensions] Ginsberg, L., P, T., Aldrin, S., and J. Zhang, "BIER support via ISIS", draft-ietf-bier-isis-extensions-01 (work in progress), October 2015.

[I-D.ietf-bier-mpls-encapsulation] Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., and S. Aldrin, "Encapsulation for Bit Index Explicit Replication in MPLS Networks", draft-ietf-bier-mpls-encapsulation-03 (work in progress), February 2016.

[I-D.ietf-bier-mvpn] Rosen, E., Sivakumar, M., Aldrin, S., Dolganow, A., and T. P, "Multicast VPN Using BIER", draft-ietf-bier-mvpn-02 (work in progress), December 2015.

[I-D.ietf-bier-ospf-bier-extensions] Psenak, P., Kumar, N., Wijnands, I., Dolganow, A., P, T., Zhang, J., and S. Aldrin, "OSPF Extensions For BIER", draft-ietf-bier-ospf-bier-extensions-01 (work in progress), October 2015.

[I-D.ietf-ospf-prefix-link-attr] Psenak, P., Gredler, H., rjs@rob.sh, r., Henderickx, W., Tantsura, J., and A. Lindem, "OSPFv2 Prefix/Link Attribute

Advertisement", draft-ietf-ospf-prefix-link-attr-13 (work in progress), August 2015.

Authors' Addresses

Cui(Linda) Wang
ZTE Corporation
No.50 Software Avenue, Yuhuatai District
Nanjing
China

Email: wang.cui1@zte.com.cn

Zheng(Sandy) Zhang
ZTE Corporation
No.50 Software Avenue, Yuhuatai District
Nanjing
China

Email: zhang.zheng@zte.com.cn

Andrew Qu
2860 Junction Ave
San JoseGBP[not] CA 95134

Email: laodulaodu@gmail.com

BIER WG
Internet-Draft
Intended status: Standards Track
Expires: July 21, 2016

C. Wang
Z. Zhang
F. Hu
ZTE Corporation

January 18, 2016

# BIER Use Case in Data Centers
# draft-wang-bier-vxlan-use-case-01

## Abstract

Bit Index Explicit Replication (BIER) is an architecture that provides optimal multicast forwarding through a "BIER domain" without requiring intermediate routers to maintain any multicast related per-flow state. BIER also does not require any explicit tree-building protocol for its operation. A multicast data packet enters a BIER domain at a "Bit-Forwarding Ingress Router" (BFIR), and leaves the BIER domain at one or more "Bit-Forwarding Egress Routers" (BFERs). The BFIR router adds a BIER header to the packet. The BIER header contains a bit-string in which each bit represents exactly one BFER to forward the packet to. The set of BFERs to which the multicast packet needs to be forwarded is expressed by setting the bits that correspond to those routers in the BIER header.

This document tries to describe the drawbacks of how BUM services are deployed in current data centers, and proposes how to take full advantage of BIER to implement BUM services in data centers.

## 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 21, 2016.

## Copyright Notice

Table of Contents

## 1. Introduction

This document is motivated by [I-D.ietf-bier-use-cases].

In current data center virtualization, virtual eXtensible Local Area Network (VXLAN) [RFC7348] is a kind of network virtualization overlay technology which is overlaid between NVEs and is intended for multi-tenancy data center networks, whose reference architecture is illustrated as per Figure 1.

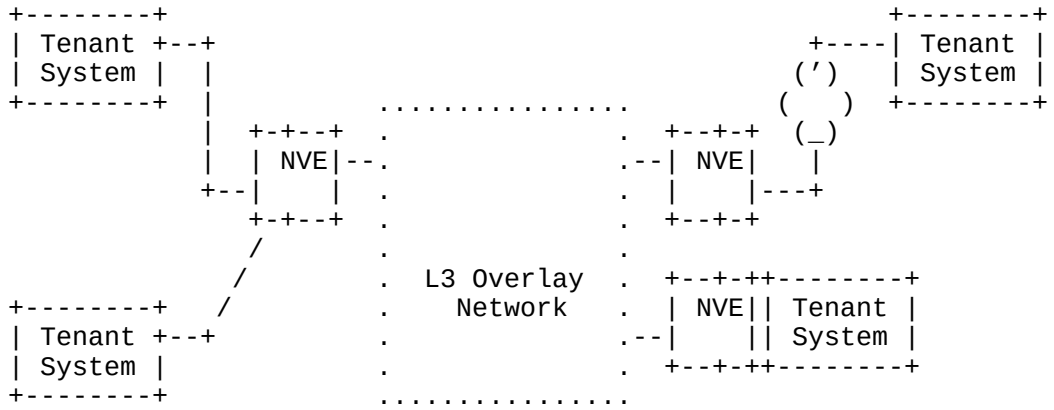



Figure 1: NVO3 Architecture

And there are two kinds of most common methods about how to forward BUM packets in this virtualization overlay network. One is using PIM as underlay multicast routing protocol to build explicit multicast distribution tree, such as PIM-SM[RFC4601] or PIM-BIDIR [RFC5015]multicast routing protocol. Then, when BUM packets arrive at NVE, it requires NVE to have a mapping between the VXLAN Network Identifier and the IP multicast group. According to the mapping, NVE can encapsulate BUM packets in a multicast packet which group address is the mapping IP multicast group address and steer them through explicit multicast distribution tree to the destination NVEs. This method has two serious drawbacks. It need the underlay network supports complicated multicast routing protocol and maintains multicast related per-flow state in every transit nodes. What is more, how to configure the ratio of the mapping between VNI and IP multicast group is also an issue. If the ratio is 1:1, there should be 16M multicast groups in the underlay network at maximum to map to the 16M VNIs, which is really a significant challenge for the data center devices. If the ratio is n:1, it would result in inefficiency

bandwidth utilization which is not optimal in data center networks.

The other method is using ingress replication to require each NVE to create a mapping between the VXLAN Network Identifier and the remote addresses of NVEs which belong to the same virtual network. When NVE receives BUM traffic from the attached tenant, NVE can encapsulate these BUM packets in unicast packets and replicate them and tunnel them to different remote NVEs respectively. Although this method can eliminate the burden of running multicast protocol in the underlay network, it has a significant disadvantage: large waste of bandwidth, especially in big-sized data center where there are many receivers.

Bit Index Explicit Replication (BIER) [I-D.ietf-bier-architecture] is an architecture that provides optimal multicast forwarding through a "BIER domain" without requiring intermediate routers to maintain any multicast related per-flow state. BIER also does not require any explicit tree-building protocol for its operation. A multicast data packet enters a BIER domain at a "Bit-Forwarding Ingress Router" (BFIR), and leaves the BIER domain at one or more "Bit-Forwarding Egress Routers" (BFERs). The BFIR router adds a BIER header to the packet. The BIER header contains a bit-string in which each bit represents exactly one BFER to forward the packet to. The set of BFERs to which the multicast packet needs to be forwarded is expressed by setting the bits that correspond to those routers in the BIER header. Specifically, for BIER-TE, the BIER header may also contain a bit-string in which each bit indicates the link the flow passes through.

The following sections try to propose how to take full advantage of overlay multicast protocol to carry virtual network information, and create a mapping between the virtual network information and the bit-string to implement BUM services in data centers.

## 2. Convention and Terminology

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

The terms about BIER are defined in [I-D.ietf-bier-architecture].

The terms about NVO3 are defined in [RFC7365].

Here tries to list the most common terminology mentioned in this draft.

BIER: Bit Index Explicit Replication(Bit Index Explicit Replication (The overall architecture of forwarding multicast using a Bit Position).

NVE: Network Virtualization Edge, which is the entity that implements the overlay functionality. An NVE resides at the boundary between a Tenant System and the overlay network.

VXLAN: Virtual eXtensible Local Area Network

VNI: VXLAN Network Identifier

Virtal Network Context Identifier: Field in an overlay encapsulation header that identifies the specific VN the packet belongs to.

## 3. BIER in data centers

This section tries to describe how to use BIER as an optimal scheme to forward the broadcast, unknown and multicast (BUM) packets when they arrive at the ingress NVE in data centers.

The principle of using BIER to forward BUM traffic is that: firstly, it requires each ingress NVE to have a mapping between the Virtual Network Context Identifier and the bit-string in which each bit represents exactly one egress NVE to forward the packet to. And then, when receiving the BUM traffic, the BFIR/Ingree NVE maps the receiving BUM traffic to the mapping bit-string, encapsulates the BIER header, and forwards the encapsulated BUM traffic into the BIER domain to the other BFERs/Egress NVEs indicated by the bit-string.

Furthermore, as for how each ingress NVE knows the other egress NVEs that belong to the same virtual network and creates the mapping is the main issue discussed below. Basically, BIER Multicast Listener Discovery is an overlay solution to support ingress routers to keep per-egress-router state to construct the BIER bit-string associated with IP multicast packets entering the BIER domain. The following section tries to extend BIER MLD to carry virtual network information(such as Virtual Network Context identifier), and advertise them between NVEs. When each NVE receive these information, they create the mapping between the virtual network information and the bit-string representing the other NVEs belonged to the same virtual network.

## 4. BIER MLD extension for Virtual Network information

Figure 2 draws the MLD report message format. In order to support Virtual Network information advertisement, one bit of the reserved field can be used to indicate that there is Virtual Network information in the message.

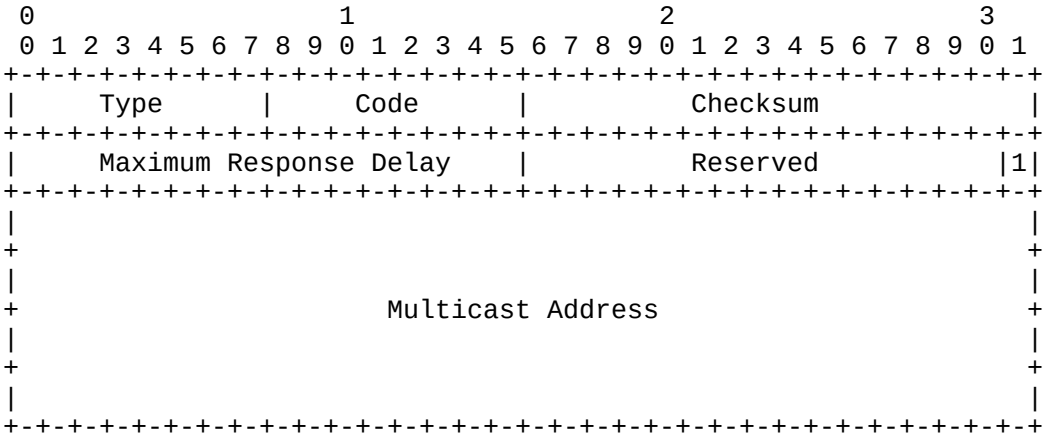



Figure 2: MLD message format

Specifically, Figure 3 illustrates the extension TLV format in MLD report message to carry Virtual Network information.

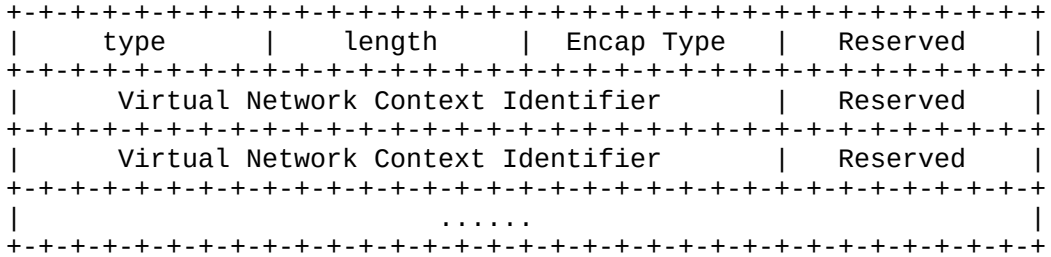



Figure 3: MLD Virtual Network information TLV

Type:

indicates Virtual Network information TLV. Value 1 indicates Virtual Network information advertisement. Value 2 indicates Virtual Network information withdraw.

Length: 1 cotet.

Encap Type:

indicates the encapsulation type the virtual netowrk using, such as VxLAN,NVGRE,Geneve and so on.

Virtual Network Context Idenfifier:

indicates the identifier of the virtual network. Different encapsulation type has different meaning for this field. In VxLAN encasulation type, this field indicates VxLAN Network Identifier. In NVGRE encapsulation type, this field indicates Virtual Subnet ID(VSID).In Geneve encapsulation type, this field indicates Virtual Network Identifier.

Each NVE acquires the Virtual Network information, and advertises this Virtual Network information to other NVEs through the MLD messages. If the NVE attaches to several virtual networks, it will carry several Virtual Network Context Identifiers in the Virtual Network advertisement message. if the NVE supports several encapsulation types, it will carry the Virtual Network Context Identifiers belonging to one encapsulation type in one Virtual Network information TLV. if one attached virtual network is migrated, the NVE will withdraw the Virtual Network information.

When ingress NVE receives the Virtual Network information advertisement message, it builds a mapping between the receiving Virtual Network Context Identifier in this message and the bit-string in which each bit represents one egress NVE who sends the same Virtual Network information. Subsequently, once this ingress NVE receives some other MLD advertisements which include the same Virtual Network information from some other NVEs , it updates the bit-string in the mapping and adds the corresponding sending NVE to the updated bit-string. Once the ingress NVE removes one virtual network, it will delete the mapping corresponding to this virtual network as well as send withdraw message to other NVEs.

After finishing the above interaction of MLD messages, each ingress NVE knows where the other egress NVEs are in the same virtual network. When receiving BUM traffic from the attached virtual network, each ingress NVE knows exactly how to encapsulate this traffic and where to forward them to.

This can be used in both IPv4 network and IPv6 network. In IPv4, IGMP protocol does the similar extension for carrying Virtual Network information TLV in Version 2 membership report message.

## 5. Security Considerations

It will be considered in a future revision.

## 6. IANA Considerations

There need one new Type for Virtual Network information TLV in MLD protocol and one in IGMP protocol.

# 7. References

## 7.1. Normative References

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

[RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, DOI 10.17487/RFC4601, August 2006, <http://www.rfc-editor.org/info/rfc4601>.

[RFC5015] Handley, M., Kouvelas, I., Speakman, T., and L. Vicisano, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)", RFC 5015, DOI 10.17487/RFC5015, October 2007, <http://www.rfc-editor.org/info/rfc5015>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <http://www.rfc-editor.org/info/rfc7348>.

[RFC7365] Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y. Rekhter, "Framework for Data Center (DC) Network Virtualization", RFC 7365, DOI 10.17487/RFC7365, October 2014, <http://www.rfc-editor.org/info/rfc7365>.

## 7.2. Informative References

[I-D.ietf-bier-architecture] Wijnands, I., Rosen, E., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast using Bit Index Explicit Replication", draft-ietf-bier-architecture-02 (work in progress), July 2015.

[I-D.ietf-bier-idr-extensions] Xu, X., Chen, M., Patel, K., Wijnands, I., and T. Przygienda, "BGP Extensions for BIER", draft-ietf-bier-idr-extensions-00 (work in progress), September 2015.

[I-D.ietf-bier-isis-extensions] Ginsberg, L., Przygienda, T., Aldrin, S., and J. Zhang,

"BIER support via ISIS",
draft-ietf-bier-isis-extensions-01 (work in progress),
October 2015.

[I-D.ietf-bier-ospf-bier-extensions]
Psenak, P., Kumar, N., Wijnands, I., Dolganow, A.,
Przygienda, T., Zhang, J., and S. Aldrin, "OSPF Extensions
For BIER", draft-ietf-bier-ospf-bier-extensions-01 (work
in progress), October 2015.

[I-D.ietf-bier-use-cases]
Kumar, N., Asati, R., Chen, M., Xu, X., Dolganow, A.,
Przygienda, T., arkadiy.gulko@thomsonreuters.com, a.,
Robinson, D., and V. Arya, "BIER Use Cases",
draft-ietf-bier-use-cases-01 (work in progress),
August 2015.

Authors' Addresses

Cui(Linda) Wang
ZTE Corporation
No.50 Software Avenue, Yuhuatai District
Nanjing
China

Email: wang.cui1@zte.com.cn

Zheng(Sandy) Zhang
ZTE Corporation
No.50 Software Avenue, Yuhuatai District
Nanjing
China

Email: zhang.zheng@zte.com.cn

Fangwei Hu
ZTE Corporation

Email: hu.fangwei@zte.com.cn

BIER WG
Internet-Draft
Intended status: Standards Track
Expires: September 7, 2016

Zheng. Zhang
Bo. Wu
Cui. Wang
ZTE Corporation

March 6, 2016

# Path Autogeneration in BIER
## draft-zhang-bier-path-autogeneration-01

Abstract

[I-D.ietf-bier-architecture] BIER introduces a method for multicast flow forwarding, without storing states in every node along the multicast path. This document introduces a method of establishing multicast path automatically.

Table of Contents



## 1. Introduction

[I-D.ietf-bier-architecture] BIER is a technology of multicast flow forwarding. [I-D.ietf-bier-mpls-encapsulation] introduces a way to use mpls in BIER forwarding. This document introduces a method of establishing multicast path automatically.

Upstream-assigned label is used in this document. Associate with the label indicated the BFIR, the two labels compose the keywords for multicast flow forwarding. Every node along the multicast path used the two labels to forward multicast flow without label changing. Obviously, the nodes along the path establish the mpls forwarding table, when they receive the packet which include the label combination and the path specification.

BFIR sends the first packet with label combination and path specification, every node along the path build the mpls forwarding table, and forward the packet to next node according to the path specification. If there is already one mpls forwarding item which has the same label combination, the node combines the next-hop in the packet to the existed forwarding item.

## 2. Packet Formats

This document introduces a new TLV that is carried by the BIER packet, and this TLV is composed by the nodes in the multicast path. There is a flag in the BIER header indicates that a path TLV is be carried. One of the reserved flag may be used to signal this.

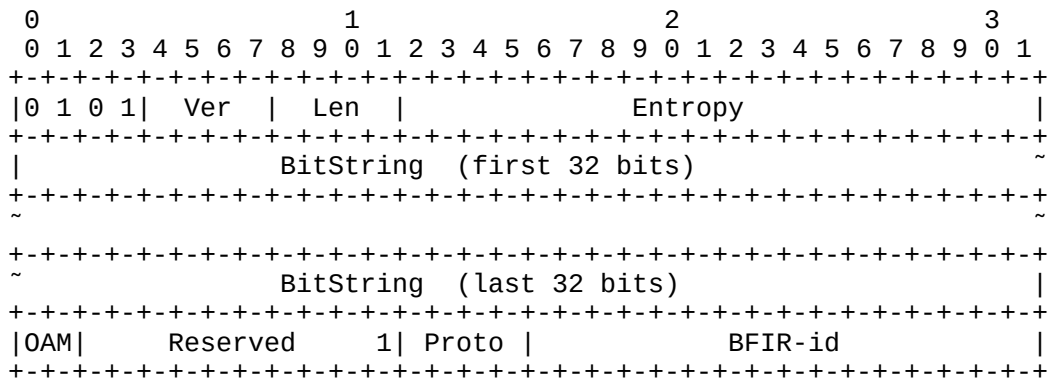



Figure 1 BIER Header

## 2.1. Node function

The path list is composed by nodes along the multicast path. The multicast path is decided by BFIR in advance through PCE or other calculations or configurations.

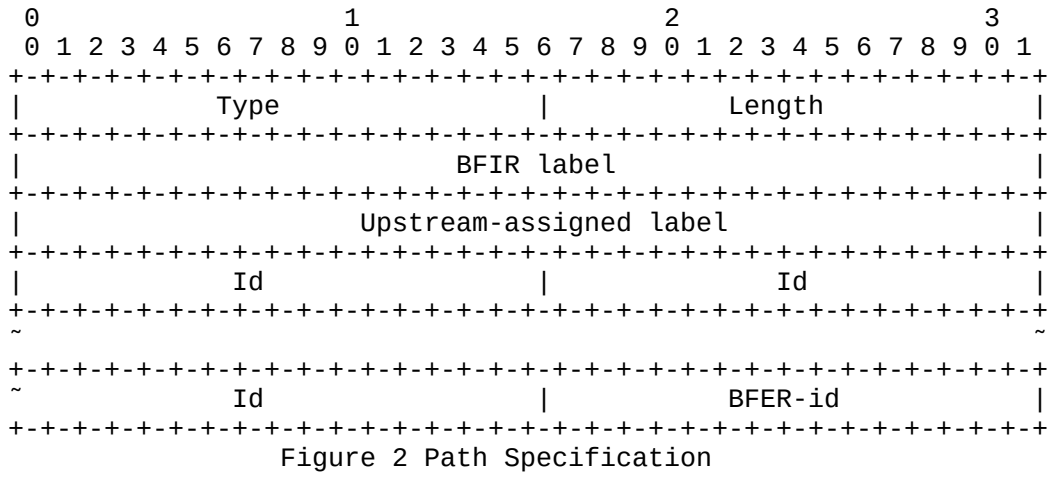



Figure 2 Path Specification

- Type TBD, indicate that there is a path TLV.
- Length The length of the TLV.
- BFIR label The label of BFIR.
- Upstream-assigned label The label that is assigned by BFIR for the specific multicast flow.

- Id The node BFR-id or the link id along the multicast path. They are carried by the packets in order.

- BFER-id The BFR-id of BFER.

Like the ingress replication, BFIR repeats BIER packets several times according to every BFER of the multicast flow. And then BFIR sends the BIER packets with the two labels.

The withdraw packet is the same format with the path packet. But the type should be another type, and the last node that is carried by the withdraw packet is the canceled node.

## 2.2. TE function

Also, the method of path auto-generation can be used in BIER-TE forwarding.

The format of TLV that is carried by BIER header is same as the previous section. And the difference is that the BitString is BIER-TE forwarding BitString. The packet is forwarding by the BIER-TE Forwarding Table.

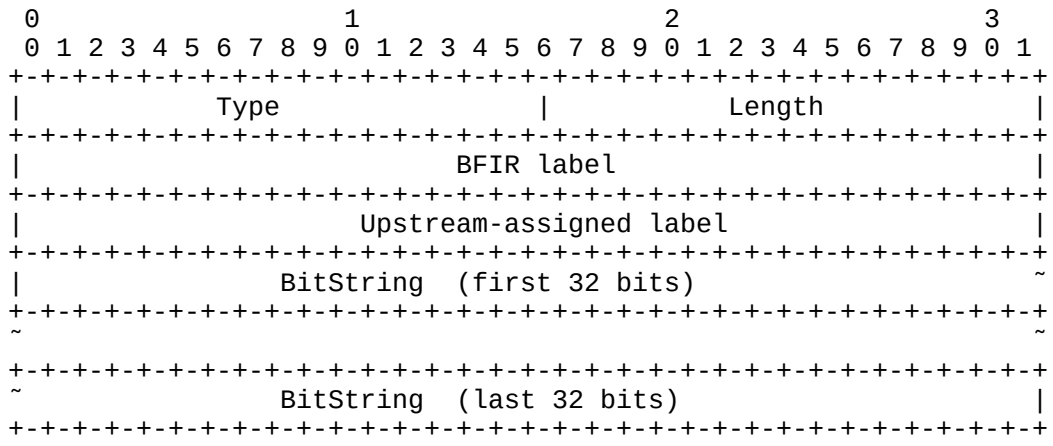



Figure 3 TE path specification

- Type TBD, indicate that there is a TE path TLV.

- Length The length of the TLV.

- BFIR label The label of BFIR.

- Upstream-assigned label The label that is assigned by BFIR for the specific multicast flow.

o BitString The adjacency of TE path, and it is the same as the BitString in the BIER header.

When the multicast flow travels the BIER-TE path by BIER-TE forwarding, the two level labels also can be established in every BFR. If some adjacency should be withdrawn from the path, the type will be another type of TE path withdraw TLV.

## 3. Procedures

### 3.1. Node procedure

#### 3.1.1. Sending the path packets

When BFIR gains the BFER information of a specific multicast flow, BFIR encapsulates the receiving flow with the path list, and sends to every BFER individually.Particularly, the encapsulated packet that is sent by BFIR includes the label combination.

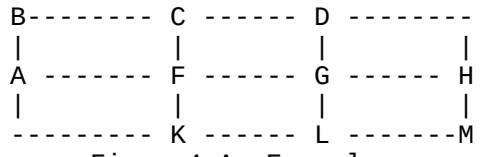



Figure4 An Example

For example, in figure 3, A is BFIR for a specific multicast flow. And D, H, M is BFER. According the PCE calculation, the multicast flow should be sent along these paths: A-F-G-D/H, A-K-L-M. So A encapsulates the flow with the two label combination and makes three copies, and then sends to D/H/M separately. In node A, there is one mpls label forwarding item which the ingress label is the two labels combination, and the next-hops are F and K with the same two labels out. A encapsulates the formal multicast flow with the two labels and forwards to next hops.

For the stability consideration, BFIR sends the path packets every once in a while. The interval may be 30 minutes. And when new BFER joined, BFIR sends the path packets to the new BFER immediately.

#### 3.1.2. Withdraw Nodes

According to the PCE calculation, some existed nodes in the path may be canceled. BFIR sends the withdraw path packet which the last node is the canceled node.

For example in figure 1, if the bandwidth in the network is changed, the multicast flow should be forwarded along these ways: A-B-C-D-H,

A-K-L-M. The specific multicast flow will not pass by the node F and G anymore. Except A sending a new path packet along the path A-B-C-D-H, A also sends two withdraw path packets to node F and G.

### 3.1.3. BFR and BFER

When BFR and BFER receive the path packets, they build an item in the mpls forwarding table according to the label combination that is carried by the packets. If there is an item already in the forwarding table, the nodes should add the next-hop which is the next node in the path packet.

When BFR and BFER receive the withdraw path packets which indicate that they should be canceled, the BFR and BFER delete the mpls forwarding item which line with the label combination. If the middle node finds that the next node that along the path is the canceled node, the middle node deletes the next-hop in the mpls forwarding item. If there is no next-hop in the mpls forwarding item, the node deletes the mpls item in the mpls forwarding table.

## 3.2. BIER-TE procedure

### 3.2.1. Sending the path packets

When BFIR gains the BFER information of a specific multicast flow, and BFIR knows all the adjacencies that the flow should travel, BFIR encapsulates the receiving flow with the BIER-TE adjacencies, and sends to BIER domain. If there are several SI should be encapsulated, the flow will be encapsulated by different BitString several times.

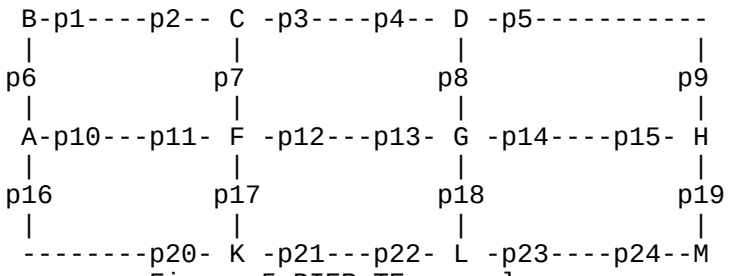



Figure 5 BIER-TE example

For example, in figure 4, A is BFIR for a specific multicast flow. And D, H, M is BFER. According the PCE calculation, the multicast flow should be sent along these paths: A-F-G-D/H, A-K-L-M. The BitPositions in BitString are p8, p10, p12, p14, p16, p21, p23. A encapsulates the flow with the two label combination and the BIER-TE BitString.

Like the node function, when the packet goes through the path of BIER-TE, every BFR establishes the label forwarding item.

### 3.2.2. Withdraw BIER-TE adjacencies

If some adjacencies in the existed path should be withdraw, the ingress node send the packet with withdraw type of TLV. The BitString in BIER header is previous TE path adjacency, and the BitString in the TLV is composed of withdraw adjacencies. When BFR receives the packet, BFR will withdraw the associated adjacencies itself.

## 4. Security Considerations

There should be some common security methods to guarantee the validation of path packets.

## 5. IANA considerations

IANA is requested to allocate four types in TLVs of BIER path packets. Two for node function and two for TE function.

## 6. Normative References

[I-D.eckert-bier-te-arch]
Eckert, T. and G. Cauchie, "Traffic Enginering for Bit Index Explicit Replication BIER-TE", draft-eckert-bier-te-arch-02 (work in progress), October 2015.

[I-D.ietf-bier-architecture]
Wijnands, I., Rosen, E., Dolganow, A., P, T., and S. Aldrin, "Multicast using Bit Index Explicit Replication", draft-ietf-bier-architecture-03 (work in progress), January 2016.

[I-D.ietf-bier-mpls-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., and S. Aldrin, "Encapsulation for Bit Index Explicit Replication in MPLS Networks", draft-ietf-bier-mpls-encapsulation-03 (work in progress), February 2016.

## Authors' Addresses

Zheng(Sandy) Zhang
ZTE Corporation
No. 50 Software Ave, Yuhuatai Distinct
Nanjing
China

Email: zhang.zheng@zte.com.cn

Bo Wu
ZTE Corporation
No. 50 Software Ave, Yuhuatai Distinct
Nanjing
China

Email: wu.bo@zte.com.cn

Cui(Linda) Wang
ZTE Corporation
No. 50 Software Ave, Yuhuatai Distinct
Nanjing
China

Email: wang.cui1@zte.com.cn

BIER WG
Internet-Draft
Intended status: Standards Track
Expires: September 1, 2016

Zheng. Zhang
Cui. Wang
Ran. Chen
Fangwei. Hu
ZTE Corporation

February 29, 2016

# BIER TE YANG module
# draft-zhang-bier-te-yang-01

## Abstract

This document defines a YANG data model for BIER TE configuration and operation.

## 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 1, 2016.

## Copyright Notice

Table of Contents



## 1. Introduction

[I-D.eckert-bier-te-arch] introduces an architecture for BIER-TE: Traffic Engineering for Bit Index Explicit Replication (BIER). This document defines a YANG data model for BIER TE. The content is in keeping with the TE architecture draft.

## 2. Design of the Data Model

Instead of using respective sub-domain-id, si and bsl information like in BIER yang draft [I-D.chh-bier-bier-yang], this document tries to group these sub-domain-id, si and bsl information in a new bier-common grouping to simplify the reference. Later yang modules may import the common grouping easily. Further, if this optimization is recognized, then BIER yang draft [I-D.chh-bier-bier-yang] will be updated to group these sub-domain-id, si and bsl information as well.

```
module: ietf-bier-te
augment /rt:routing:
   +--rw bier-te-config
      +--rw te-subdomain* [subdomain-id]
         +--rw subdomain-id    sub-domain-id
         +--rw adj-id* [adjID]
         |  +--rw adjID     adjid
         |  +--rw adj-if    uint32
         |  +--rw (te-adjID-type)
         |     +--:(p2p)
         |     +--:(bfer)
         |     +--:(leaf-bfer)
         |     +--:(lan)
         |     +--:(spoke)
         |     +--:(ring-clockwise)
         |     +--:(ring-counterclockwise)
         |     +--:(ecmp)
         |     +--:(virtual-link)
         |     +--:(other)
         +--rw te-bsl* [fwd-bsl]
         |  +--rw fwd-bsl    uint16
```

```
                     |  +--rw te-si* [si]
                     |     +--rw si            si
                     |     +--rw te-f-index* [te-f-index]
                     |        +--rw te-f-index    bit-string
                     |        +--rw (te-adj-type)
                     |        |  +--:(connected)
                     |        |  +--:(routed)
                     |        |  +--:(local-decap)
                     |        |  +--:(ecmp)
                     |        |  +--:(other)
                     |        +--rw f-bm          bit-string
                     |        +--rw f-intf        uint32
                     |        +--rw ecmp?         boolean
                     |        +--rw frr?          boolean
                     +--rw ecmp-path* [index]
                     |  +--rw index     uint32
                     |  +--rw number* [number]
                     |     +--rw number    uint16
                     |     +--rw out-if    uint32
                     +--rw btaft* [adj-index]
                        +--rw adj-index       uint32
                        +--rw bitposition     bit-string
                        +--rw resetbitmask    bit-string
                        +--rw addbitmask      bit-string
               augment /rt:routing:
                  +--ro bier-te-state
                     +--ro te-subdomain* [subdomain-id]
                        +--ro subdomain-id    sub-domain-id
                        +--ro adj-id* [adjID]
                        |  +--ro adjID     adjid
                        |  +--ro adj-if    uint32
                        |  +--ro (te-adjID-type)
                        |     +--:(p2p)
                        |     +--:(bfer)
                        |     +--:(leaf-bfer)
                        |     +--:(lan)
                        |     +--:(spoke)
                        |     +--:(ring-clockwise)
                        |     +--:(ring-counterclockwise)
                        |     +--:(ecmp)
                        |     +--:(virtual-link)
                        |     +--:(other)
                        +--ro te-bsl* [fwd-bsl]
                        |  +--ro fwd-bsl    uint16
                        |  +--ro te-si* [si]
                        |     +--ro si            si
                        |     +--ro te-f-index* [te-f-index]
                        |        +--ro te-f-index    bit-string
```

```
                     |        +--ro (te-adj-type)
                     |        |  +--:(connected)
                     |        |  +--:(routed)
                     |        |  +--:(local-decap)
                     |        |  +--:(ecmp)
                     |        |  +--:(other)
                     |        +--ro f-bm          bit-string
                     |        +--ro f-intf        uint32
                     |        +--ro ecmp?         boolean
                     |        +--ro frr?          boolean
                     +--ro ecmp-path* [index]
                     |  +--ro index     uint32
                     |  +--ro number* [number]
                     |     +--ro number    uint16
                     |     +--ro out-if    uint32
                     +--ro btaft* [adj-index]
                        +--ro adj-index       uint32
                        +--ro bitposition     bit-string
                        +--ro resetbitmask    bit-string
                        +--ro addbitmask      bit-string
               notifications:
                  +---n bier-te-notification
                     +--ro adjID-is-zero* [if-index]
                        +--ro if-index    uint32
                        +--ro (te-adjID-type)
                           +--:(p2p)
                           +--:(bfer)
                           +--:(leaf-bfer)
                           +--:(lan)
                           +--:(spoke)
                           +--:(ring-clockwise)
                           +--:(ring-counterclockwise)
                           +--:(ecmp)
                           +--:(virtual-link)
                           +--:(other)
```

## 3. BIER-TE configuration

The BIER-TE information is indexed by the sub-domain ID. Maybe there are some global BIER-TE information, it should be added in later version.

One interface can be used in different sub-domain, so the BIER TE adjacency information is managed by BIER TE other than by interface.

Because the BIER-TE is controlled by controller now, the information about IGP is not defined. If in the future the IGP is used to carry

the information about BIER-TE, the IGP extension will be added in this document.

## 4. Notifications

If the adjacency id of one adjacency is set to zero, the value is invalid. The notification should be sent to controller and network manager.

## 5. BIER TE YANG module

<CODE BEGINS> file "ietf-bier-te.yang"

```
module ietf-bier-te {

 namespace "urn:ietf:params:xml:ns:yang:ietf-bier-te";

 prefix bier-te;

 import ietf-routing {
     prefix "rt";
 }
 /* import bier-common {
     prefix "bier-common";
 } */

 organization " IETF BIER(Bit Indexed Explicit Replication ) Working Group";
 contact
     "WG List:  <mailto:bier@ietf.org>
      WG Chair: Tony Przygienda
                <mailto:tonysietf@gmail.com>
      WG Chair: Greg Shepherd
                <mailto:gjshep@gmail.com>

      Editor:   Zheng Zhang
                <mailto:zhang.zheng@zte.com.cn>
      Editor:   Cui Wang
                <mailto:wang.cui1@zte.com.cn>
      Editor:   Ran Chen
                <mailto:chen.ran@zte.com.cn>
      Editor:   Fangwei Hu
                <mailto:hu.fangwei@zte.com.cn>
     ";

 description
     "This module contains a collection of YANG definitions for
      managing BIER TE information.";

 revision 2016-03-01 {
```

```
        description
        "Initial version.";
        reference "https://tools.ietf.org/html/draft-zhang-bier-te-yang";
    }

        grouping te-adj-type {
        description "The collection of all possible adjacency type.";
        choice te-adj-type {
            mandatory true;
            case connected {
                description "The type of adjacency is connected. Mostly connecte
d interfaces.";
                        }
            case routed {
                description "The type of adjacency is routed. Mostly not connect
ed interfaces.";
            }
            case local-decap {
                description "Means that the packet should be decapsulated and fo
rward out BIER domain.";
            }
            case ecmp {
                description "There is more than one path in the adjacency with e
qual cost.";
            }
            case other {
                description "Means that the packet should be discarded.";
            }
            description "The collection of all possible adjacency type.";
        }
    }

    grouping te-adjID-type {
        description "The collection of all possible adjacency type.";
        choice te-adjID-type {
            mandatory true;
            case p2p {
                description "Describes p2p adjacency.";
            }
            case bfer {
                description "Describes bfer adjacency.";
            }
            case leaf-bfer {
                description "Describes leaf-bfer adjacency. There is no next BFR
 that the packet should be forwarded.";
            }
            case lan {
                description "Describes lan adjacency..";
            }
            case spoke {
                description "Describes spoke adjacency of hub-and-spoke.";
            }
            case ring-clockwise {
```

```
                description "Describes clockwise adjacency in ring.";
            }
            case ring-counterclockwise {
                description "Describes counterclockwise adjacency in ring.";
            }
            case ecmp {
                description "Describes  ecmp adjacency.";
            }
            case virtual-link {
                description "Describes virtual adjacency between two indirect co
nnect nodes.";
            }
            case other {
                description "Describes other id type of adjacency.";
            }
            description "The collection of all possible adjacency type.";
        }
    }

        typedef bsl {
        type enumeration{
            enum 64-bit{
                description "bitstringlength is 64";
            }
            enum 128-bit{
                description "bitstringlength is 128";
            }
            enum 256-bit{
                description "bitstringlength is 256";
            }
            enum 512-bit{
                description "bitstringlength is 512";
            }
            enum 1024-bit{
                description "bitstringlength is 1024";
            }
            enum 2048-bit{
                description "bitstringlength is 2048";
            }
            enum 4096-bit{
                description "bitstringlength is 4096";
            }
        }
        description "The bitstringlength type for imposition mode.";
    }

    typedef adjid {
        type uint32;
        description "The type for adjacency ID.";
```

```
    }

    /* The definition of si/sub-domain-id/bit-string will be deleted later. */
    typedef si {
        type uint16;
        description "The type for set identifier";
    }

    typedef sub-domain-id {
        type uint16;
        description "The type for sub-domain-id";
    }

    typedef bit-string {
        type uint16;
        description "The bit mask of one bitstring.";
    }

    /* The bier-common grouping will be moved to BIER yang. */
    grouping bier-common {
        description "Common information in BIER";
        leaf subdomain-id {
            type sub-domain-id;
            description "ID of the sub domain.";
        }
        leaf si {
            type si;
            description "The value of the set identifier.";
        }
        leaf bsl {
            type bsl;
            description "The BitStringLength supported by this node.";
        }
        leaf bit-string {
            type bit-string;
            description "The bit-string of BIER forwarding.";
        }
    }

        grouping te-adjID {
        list adj-id {
            key "adjID";
            description "This ID information of one adjacency.";
            leaf adjID {
                type adjid;
                mandatory true;
                description "The adjacency id.";
            }
```

```
            leaf adj-if {
                /* type if:if-index; */
                type uint32;  /* for compilation */
                mandatory true;
                description "The corresponding interface of this adjacency.";
            }
            uses te-adjID-type;
        }
        description "This group presents adjacency ID information for BIER TE.";
    }

    grouping te-ecmp {
        description "The ecmp information.";
        list ecmp-path {
            key "index";
            description "The index of the ecmp paths.";
            leaf index {
                type uint32;
                mandatory true;
                description "The ecmp index.";
            }
            list number {
                key "number";
                description "The list of the ecmp paths.";
                leaf number {
                    type uint16;
                    mandatory true;
                    description "The number of the ecmp paths.";
                }
                leaf out-if {
                    /* type if:if-index; */
                    type uint32; /* for compilation */
                    mandatory true;
                    description "The outgoing interface.";
                }
            }
        }
    }

    grouping te-frr {
        description "The TE fast reroute information.";
        list btaft {
            key "adj-index";
            description "The adjacency index of the frr paths.";
            leaf adj-index {
                type uint32;
                mandatory true;
                description "The frr adjacency index.";
```

```
            }
            leaf bitposition {
                type bit-string;
                mandatory true;
                description "The bitposition information.";
            }
            leaf resetbitmask {
                type bit-string;
                mandatory true;
                description "The deleting bitmask of the forwarding item.";
            }
            leaf addbitmask {
                type bit-string;
                mandatory true;
                description "The adding bitmask of the forwarding item.";
            }
        }
    }

    grouping te-items {
        description "The BIER TE forwarding items collection.";
        uses te-adj-type;
        leaf f-bm {
            type bit-string;
            mandatory true;
            description "The bitmask of the forwarding item.";
        }
        leaf f-intf {
            /* type if:if-index; */
            type uint32; /* for compilation */
            mandatory true;
            description "The out interface of this forwarding item.";
        }
        leaf ecmp {
            type boolean;
            description "The capibility of ecmp paths.";
        }
        leaf frr {
            type boolean;
            description "The capibility of fast re-route.";
        }
    }

    grouping te-fwd-item {
        list te-si {
            key "si";
            description "The forwarding items of one set identifier.";
            leaf si{
```

```
                type si;
                    mandatory true;
                description "The set identifier of this forwarding item.";
            }
            list te-f-index {
                key "te-f-index";
                description "The forwarding information of one BIER TE item.";
                leaf te-f-index {
                    type bit-string;
                                mandatory true;
                    description "The bit index of BIER TE forwarding item.";
                }
                uses te-items;
            }
        }
        description "The forwarding items in one set identifier.";
    }

    grouping te-info {
        description "The BIER TE forwarding information.";
        list te-subdomain {
            key "subdomain-id";
            description "The forwarding items of one sub-domain.";
            leaf subdomain-id {
                type sub-domain-id;
                description "The sub-domain-id of this sub-domain.";
            }
            uses te-adjID;

            list te-bsl {
                key "fwd-bsl";
                description "The forwarding items in one bitstringlength.";
                leaf fwd-bsl {
                    type uint16;
                    description "The value of bitstringlength.";
                }
                uses te-fwd-item;
            }
            uses te-ecmp;
            uses te-frr;
        }
    }

    augment "/rt:routing" {
        description "The BIER TE information.";
        container bier-te-config {
            description "The BIER TE information container.";
            uses te-info;
```

```
        }
    }

    augment "/rt:routing" {
        description "The read-only BIER TE information.";
        container bier-te-state {
            config false;
            description "The BIER TE information in nodes.";
            uses te-info;
        }
    }

    notification bier-te-notification {
        description
          "This notification is sent when a condition changes in BIER TE.";
        list adjID-is-zero {
            key "if-index";
            description "The adjacency id is zero.";
            leaf if-index {
                type uint32;
                description "The adjacency id of this interface is zero.";
            }
            uses te-adjID-type;
        }
    }
}
<CODE ENDS>
```

## 6. Normative References


[I-D.chh-bier-bier-yang]
Chen, R., hu, f., Zhang, Z., dai.xianxian@zte.com.cn, d., and M. Sivakumar, "YANG Data Model for BIER Protocol", draft-chh-bier-bier-yang-02 (work in progress), November 2015.

[I-D.eckert-bier-te-arch]
Eckert, T. and G. Cauchie, "Traffic Enginering for Bit Index Explicit Replication BIER-TE", draft-eckert-bier-te-arch-02 (work in progress), October 2015.

[I-D.ietf-bier-architecture]
Wijnands, I., Rosen, E., Dolganow, A., P, T., and S. Aldrin, "Multicast using Bit Index Explicit Replication", draft-ietf-bier-architecture-03 (work in progress), January 2016.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, DOI 10.17487/RFC6087, January 2011, <http://www.rfc-editor.org/info/rfc6087>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <http://www.rfc-editor.org/info/rfc7223>.

## Authors' Addresses

Zheng(Sandy) Zhang
ZTE Corporation
No. 50 Software Ave, Yuhuatai Distinct
Nanjing
China

Email: zhang.zheng@zte.com.cn

Cui(Linda) Wang
ZTE Corporation
No. 50 Software Ave, Yuhuatai Distinct
Nanjing
China

Email: wang.cui1@zte.com.cn

Ran Chen
ZTE Corporation
No. 50 Software Ave, Yuhuatai Distinct
Nanjing
China

Email: chen.ran@zte.com.cn

Fangwei Hu
ZTE Corporation
No.889 Bibo Rd
Shanghai
China

Email: hu.fangwei@zte.com.cn