Networking Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 15, 2016

Ran. Chen
Zheng. Zhang
ZTE Corporation

October 13, 2015

PCEP Extensions for BIER
draft-chen-pce-bier-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 forwards and replicates packets based on a BitString in the packet header.A BIER Path can be derived from a variety of mechanisms, including an IGP Shortest Path Tree (SPT), explicit configuration, or a Path Computation Element (PCE).

This document specifies extensions to the Path Computation Element Protocol (PCEP) to handle requests and responses for the computation of paths for BIER TE LSPs.

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 April 15, 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 forwards and replicates packets based on a BitString in the packet header.A BIER Path can be derived from a variety of mechanisms, including an IGP Shortest Path Tree (SPT), explicit configuration, or a Path Computation Element (PCE).

This document specifies extensions to the Path Computation Element Protocol (PCEP) to handle requests and responses for the computation of paths for BIER TE LSPs.

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. Overview of PCEP Operation in BIER Networks

BIER forwards and replicates packets based on a BitString in the packet header. In a PCEP session, An ERO object specified in [RFC5440] carrying a BIER-TE path consists of one or more BIER-subobject(s). BIER-TE LSPs computed by a PCE can be represented in one of the following forms:

- An ordered set of BitString(s) in which each bit represents exactly one router that the BIER-TE paths pass through in the domain.
- An ordered set of BFR-id(s). In this case, the PCC needs to convert the BFR-ID into the corresponding BitString(s).
- An ordered set of BFR-prefix(es). In this case, the PCC needs to convert the BFR-ID into the corresponding B BitString(s).

In this document, we define a set of PCEP protocol extensions, including a new PCEP capability,a new Path Setup Type (PST) ,a new BIER END-POINT Object, new ERO subobjects, new RRO subobjects, new PCEP error codes and procedures.

4. BIER PCEP Message Extensions

The following section describes the protocol extensions required to support BIER-TE path.

4.1. BIER Capability Advertisement

4.1.1. The OPEN Object

This document defines a new optional TLV for use in the OPEN Object.

4.1.1.1. The BIER PCE Capability TLV

The BIER-PCE-CAPABILITY TLV is an optional TLV associated with the OPEN Object to exchange BIER capability of PCEP speakers. The format of the BIER-PCE-CAPABILITY TLV is shown in the following figure:

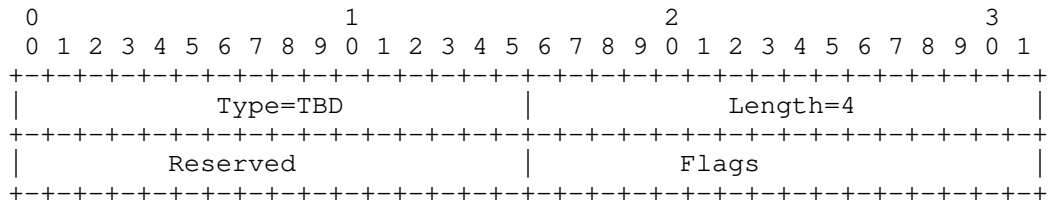


Figure 1

The code point for the TLV type is to be defined by IANA.

The "Reserved" (2 octets) and "Flags" (2 octet) fields are currently unused, and MUST be set to zero on transmission and ignored on reception.

4.1.1.1.1. Exchanging BIER Capability

This document defines a new optional BIER-PCE-CAPABILITY TLV for use in the OPEN object to negotiate the BIER capability. The inclusion of this TLV in the OPEN message destined to a PCC indicates the PCE's capability to perform BIER-TE path computations, and the inclusion of this TLV in the OPEN message destined to a PCE indicates the PCC's capability to support BIER-TE Path.

A PCE that is able to support the BIER extensions defined in this document SHOULD include the BIER-PCE-CAPABILITY TLV on the OPEN message. If the PCE does not include the BIER-PCE-CAPABILITY TLV in the OPEN message and PCC does include the TLV, it is RECOMMENDED that the PCC indicates a mismatch of capabilities.

4.2. Path Computation Request/Reply Message Extensions

4.2.1. The RP/SPR Object

In order to setup an BIER-TE LSP, a new PATH-SETUP-TYPE TLV([I-D.ietf-pce-lsp-setup-type]) MUST be contained in RP or SRP object. This document defines a new Path Setup Type (PST) for BIER as follows:

o PST = 2: Path is setup using BIER Traffic Engineering technique.

If a PCEP speaker does not recognize the PATH-SETUP-TYPE TLV, it MUST ignore the TLV in accordance with [RFC5440]. If a PCEP speaker recognizes the TLV but does not support the TLV, it MUST send PCErr with Error-Type = 2 (Capability not supported).

4.2.2. The New BIER END-POINT Object

The END-POINTS object is used in a PCReq message to specify the BIER information of the path for which a path computation is requested. To represent the end points for a BIER path efficiently, we define a new END-POINT Object for the BIER path:

The format of the new END-POINTS Object is as follows:

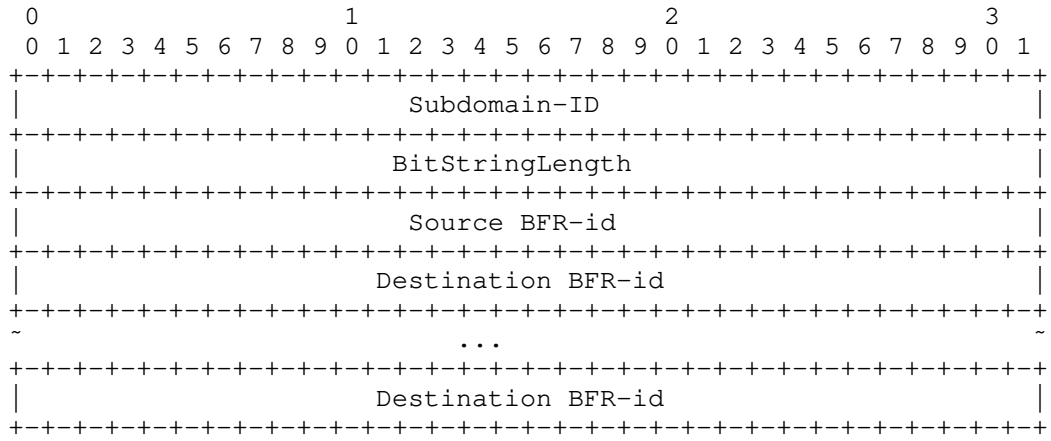


Figure 2

4.2.3. ERO Object

An BIER-TE path consists of one or more BitString/BFR-id/BFR-prefix where each BitString/BFR-id/BFR-prefix MAY be associated with the identifier that represents the node.

The ERO object specified in [RFC5440] is used to carry BIER-TE path information. In order to carry BitString/BFR-id/BFR-prefix, this document defines three new ERO subobjects referred to as "BIER-ERO subobjects" whose formats are specified in the following section. An ERO object carrying a BIER-TE path consists of one or more BIER-ERO subobject(s).

4.2.3.1. BIER-ERO Subobject

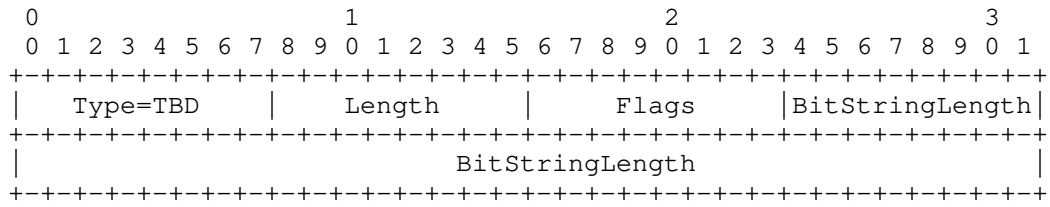


Figure 3

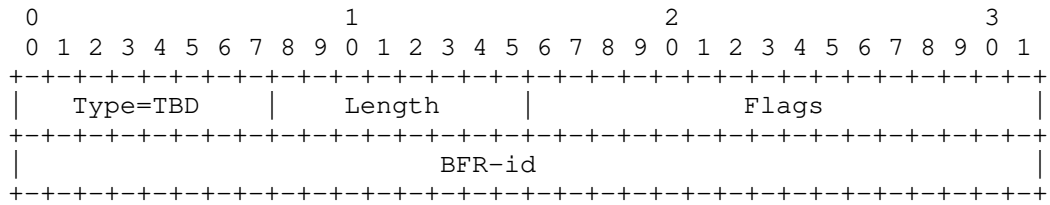


Figure 4

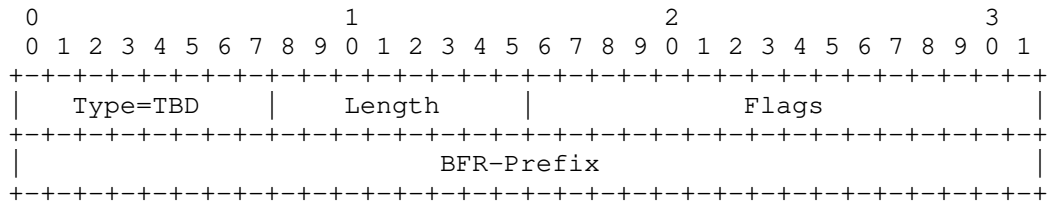


Figure 5

4.2.3.1.1. ERO Processing

If a PCC finds a non-recognize the SR-ERO subobject, the PCC MUST respond with a PCErr message with Error-Type=3 ("Unknown Object") and Error-Value=2 ("Unrecognized object Type") or Error-Type=4 ("Not supported object") and Error-Value=2 ("Not supported object Type") as described in [RFC5440] .

If a PCC receives an BIER-ERO subobject format 1 in which either BitStringLength or BitString is absent, it MUST consider the entire

ERO object invalid and send a PCErr message with Error-Type = 10 ("Reception of an invalid object") and Error-Value = TBD ("BitStringLength is absent ") and Error-Value = TBD ("BitString is absent ")

If a PCC detects that all subobjects of ERO are not identical, it MUST send a PCErr message with Error-Type = 10 ("Reception of an invalid object") and Error-Value = TBD ("Non-identical ERO subobjects"), as described in ([I-D.ietf-pce-segment-routing])[pce-].

If a PCC receives an SR-ERO subobject in which BFR-ID=0, it MUST send a PCErr message with Error-Type = 10 ("Reception of an invalid object") and Error-Value = TBD ("Invalid BFR-ID").

If a PCC receives an SR-ERO subobject in which BitStringLength values are not chosen from: 64, 128, 256, 512, 1024, 2048, and 4096, as it described in ([I-D.ietf-bier-architecture]). The PCC MUST send a PCErr message with Error-Type = 10 ("Reception of an invalid object") and Error-Value = TBD ("Invalid BitStringLength").

4.2.4. RRO Object

A PCC can record BIER-TE LSP and report the LSP to a PCE via RRO. An RRO object contains one or more subobjects called "BIER-RRO subobjects" whose formats are the same as that of SR-ERO subobject.

4.2.4.1. RRO Processing

Processing rules of SR-RRO subobject are identical to those of SR-ERO subobject defined in section 4.2.3.1.1 in this document.

5. Security Considerations

TBD.

6. IANA Considerations

6.1. PCEP Objects

As discussed in Section 4.2.2, a new END-POINTS Object-Type is defined. IANA has made the following Object-Type allocations from the "PCEP Objects" sub-registry:

Object	Object-Class Value
BIER END-POINT Object	TBD

As discussed in Section 4.2.3 and 4.2.4, a new sub-object type for the PCEP explicit route object (ERO), and a new sub-object type for the PCEP record route object (RRO) are defined.

IANA has made the following sub-objects allocation from the RSVP Parameters registry:

Object	Sub-Object	Sub-Object Type
EXPLICIT_ROUTE	BIER-ERO (PCEP-specific)	TBD
ROUTE_RECORD	BIER-RRO (PCEP-specific)	TBD

6.2. PCEP-Error Objects and Types

As described in Section 4.2.3.1.1, a number of new PCEP-ERROR Object Error Values have been defined.

Error-Type	Meaning	Reference
10	Reception of an invalid object.	RFC5540
	Error-value = TBD: BitStringLength is absent	This document
	Error-value = TBD: BitString is absent	This document
	Error-value = TBD: invalid BFR-ID	This document
	Error-value = TBD: Invalid BitStringLength	This document

6.3. PCEP TLV Type Indicators

IANA is requested to allocate a new code point in the PCEP TLV Type Indicators registry, as follows:

Value	Meaning	Reference
TBD	BIER-PCE-CAPABILITY TLV	This document

6.4. New Path Setup Type

IANA is requested to allocate a new code point in the PCEP PATH_SETUP_TYPE TLV PST field registry, as follows:

Value	Description	Reference
2	Path is setup using BIER Traffic Engineering technique	This document

7. Normative 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-pce-lsp-setup-type]
Sivabalan, S., Medved, J., Minei, I., Crabbe, E., Varga, R., Tantsura, J., and J. Hardwick, "Conveying path setup type in PCEP messages", draft-ietf-pce-lsp-setup-type-03 (work in progress), June 2015.

[I-D.ietf-pce-segment-routing]
Sivabalan, S., Medved, J., Filsfils, C., Crabbe, E., Lopez, V., Tantsura, J., Henderickx, W., and J. Hardwick, "PCEP Extensions for Segment Routing", draft-ietf-pce-segment-routing-06 (work in progress), August 2015.

[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>.

[RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <http://www.rfc-editor.org/info/rfc5440>.

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

BIER WG
Internet-Draft
Intended status: Standards Track
Expires: April 18, 2016

Ran. Chen
Fangwei. Hu
Zheng. Zhang
Xianxia. Dai
ZTE Corporation
Mahesh Sivakumar
Cisco Systems, Inc.
October 16, 2015

YANG Data Model for BIER Protocol
draft-chh-bier-bier-yang-01.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 April 18, 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       enumeration
         +--rw bfr-id                uint16
         +--rw af
         |  +--rw ipv4
         |  |  +--rw ipv4-bfr-prefix    inet:ipv4-prefix
         |  +--rw ipv6
         |     +--rw ipv6-bfr-prefix    inet:ipv6-prefix
         +--rw sub-domain* [sub-domain-id]
            +--rw sub-domain-id      uint16
            +--rw mt-id              uint16
            +--rw bfr-id?            uint16
            +--rw frr?               boolean
            +--rw bitstringlength    enumeration
            +--rw af
               +--rw ipv4* [bitstringlength bier-mpls-label]
               |  +--rw bitstringlength               uint16
               |  +--rw bier-mpls-label               uint32
               |  +--rw bier-mpls-label-range-size?   uint8
```

```
                  +--rw ipv6* [bitstringlength bier-mpls-label]
                     +--rw bitstringlength               uint16
                     +--rw bier-mpls-label               uint32
                     +--rw bier-mpls-label-range-size?   uint8

   augment /rt:routing/rt:routing-instance/rt:routing-protocols
   /rt:routing-protocol/isis:isis/isis:instance:
      +--rw mt
         +--rw mt-id          uint16
         +--rw bier-global
            +--rw enable?      boolean
            +--rw advertise?   boolean
            +--rw receive?     boolean

   augment /rt:routing/rt:routing-instance/rt:routing-protocols
   /rt:routing-protocol/ospf:ospf/ospf:instance:
      +--rw mt
         +--rw mt-id          uint16
         +--rw bier-global
            +--rw enable?      boolean
            +--rw advertise?   boolean
            +--rw receive?     boolean

   augment /rt:routing-state:
      +--ro bier-global
      |  +--ro bfr-id             uint16
      |  +--ro ipv4-bfr-prefix?   inet:ipv4-prefix
      |  +--ro ipv6-bfr-prefix?   inet:ipv6-prefix
      |  +--ro sub-domain* [sub-domain-id]
      |     +--ro sub-domain-id      uint16
      |     +--ro mt-id              uint16
      |     +--ro bfr-id?            uint16
      |     +--ro bitstringlength    uint16
      |     +--ro ipv4* [bitstringlength label]
      |     |  +--ro bitstringlength     uint16
      |     |  +--ro label?              uint32
      |     |  +--ro label-range-size?   uint8
      |     +--ro ipv6* [bitstringlength label]
      |        +--ro bitstringlength     uint16
      |        +--ro label?              uint32
      |        +--ro label-range-size?   uint8
      +--ro birts
         +--ro birt* [sub-domain-id bitstringlength si]
            +--ro sub-domain-id          uint16
            +--ro bitstringlength        uint16
            +--ro si                     uint16
            +--ro f-bm?                  uint16
            +--ro bier-mpls-in-label     uint32
```

```
            +--ro bfr-nbr?               union
            +--ro bier-mpls-out-label    uint32


      notifications:
         +---n bfr-id-collision
         |  +--ro bfr-id?   uint16
         +---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?   uint16
            +--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 transport type, imposition BitStringLengths, BFR-id, AF, 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 routing-protocol 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 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

```
 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";
     }


     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 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.";
     }

     /* Configuration Data */
    augment "/rt:routing" {
                description
                        "This augments routing-instance configuration with bier.
";
        container bier{
                description "BIER config.";
                container bier-global {
                        description
                                "BIER global config.";
                        leaf encapsulation-type {
                                type identityref {
                                        base bier-encapsulation;
                                }
                                default "bier-encapsulation-mpls";
                                description
                                        "Dataplane to be used.";
                        }
                        leaf bitstringlength{
```

```
                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
                        "imposition";
        }
        leaf bfr-id {
                type uint16;
                        description
                        "BIER bfr identifier.";
        }
        container af {
                container ipv4 {
                        leaf ipv4-bfr-prefix {
                                type inet:ipv4-prefix;
                                description
                                        "BIER IPv4 prefix.";
                        }
                        description
                                "IPv4 prefix.";
                }
                container ipv6 {
                        leaf ipv6-bfr-prefix {
                                type  inet:ipv6-prefix;
                                description
                                        "BIER IPv6 prefix.";
                        }
                        description
```

```
                                "IPv6 prefix.";
                        }
                    description
                                "BIER IPV4&IPV6 prefix";
                    }
                    list sub-domain {
                            key "sub-domain-id";
                            leaf sub-domain-id{
                                    type uint16;
                                    description
                         "sub-domain ID.";
                            }
                            leaf mt-id {
                                    type uint16;
                            description
                              "multi-topology ID.";
                            }
                            leaf bfr-id{
                                    type uint16;
                                     description
                               "BIER bfr identifier.";
                            }
                            leaf frr{
                                    type boolean;
                                    description
                              "Enables BIER FRR.";
                            }
                            leaf bitstringlength{
                                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
                                                "disposition";
                                }
                                container af {
                                        list ipv4 {
                                                key "bitstringlength bier-mpls-l
abel";
                                                leaf bitstringlength {
                                                        type uint16;
                                                    description
                                                 "BIER bitstringlength.";
                                                }
                                                leaf bier-mpls-label{
                                                        type uint32;
                                                    description
                                                 "BIER label.";
                                                }
                                                leaf bier-mpls-label-range-size{
                                                        type uint8;
                                                    description
                                                 "BIER label range.";
                                                }
                                                description
                                                        "IPv4 mapping entries.";
                                        }
                                        list ipv6 {
                                                key "bitstringlength bier-mpls-l
abel";
                                                leaf bitstringlength {
                                                        type uint16;
                                                    description
                                                 "BIER bitstringlength.";
                                                }
                                                leaf bier-mpls-label{
                                                        type uint32;
                                                    description
                                                 "BIER label.";
                                                }
                                                leaf bier-mpls-label-range-size{
                                                        type uint8;
                                                    description
                                                 "BIER label range.";
                                                }
                                                description
                                                        "IPv6 mapping entries.";
                                        }
                                description
```

```
                                "Bier mapping entries.";
                            }
                        description
                          "Parameters associated with bier sub-domain.";
                        }
                }
        }
        }
     augment "/rt:routing/rt:routing-instance/rt:routing-protocols/"
           + "rt:routing-protocol" {
                description
                        "This augments ospf protocol configuration with bier.";
                container mt{
                        description
                                "Control of bier advertisement and reception.";
                        leaf mt-id{
                                type uint16;
                                description
                                        "Multi-topology associated with bier sub
-domain.";
                        }
                        container bier-global {
                                description
                                        "BIER global config.";
                                leaf enable {
                                        type boolean;
                                        default false;
                                        description
                                                "Enables bier protocol extension
s.";
                                }
                                leaf advertise {
                                        type boolean;
                                        default true;
                                        description
                                                "Enable to advertise the paramet
ers associated with
                                                bier.";
                                }
                                leaf receive {
                                        type boolean;
                                        default true;
                                        description
                                                "Enable to receive the parameter
s associated with
                                                bier.";
                                }
                        }
                }
    }

        augment "/rt:routing/rt:routing-instance/rt:routing-protocols/"
```

```
           + "rt:routing-protocol" {
                description
                        "This augments ISIS protocol configuration with bier.";
                container mt{
                        description
                                "Control of bier advertisement and reception.";
                        leaf mt-id{
                                type uint16;
                                description
                                        "Multi-topology associated with bier sub
-domain.";
                        }
                        container bier-global {
                                description
                                        "BIER global config.";
                                leaf enable {
                                        type boolean;
                                        default false;
                                        description
                                                "Enables bier protocol extension
s.";
                                }
                                leaf advertise {
                                        type boolean;
                                        default true;
                                        description
                                                "Enable to advertise the paramet
ers associated with
                                                bier.";
                                }
                                leaf receive {
                                        type boolean;
                                        default true;
                                        description
                                                "Enable to receive the parameter
s associated with
                                                bier.";
                                }
                        }
                }
    }

         /* Operational  data */
    augment "/rt:routing-state" {
        description
                        "This augments the operational states with bier.";
                container bier-global{
                        leaf bfr-id{
                type uint16;
                                description
                                        "BIER BFR ID.";
           }
```

```
                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 uint16;
                                description
                                        "sub-domain ID.";
                        }
                        leaf mt-id {
                                type uint16;
                                description
                                 "Multi-topology ID";
                        }
                        leaf bfr-id {
                                type uint16;
                                description
                                        "BIER bfr identifier.";
                        }
                        leaf bitstringlength{
                                type uint16;
                                description
                          "BIER bitstringlength.";
                        }
                        list ipv4 {
                                key "bitstringlength label";
                                leaf bitstringlength {
                                        type uint16;
                                        description
                              "BIER bitstringlength.";
                                }
                                leaf label{
                                        type uint32;
                                        description
                                  "BIER label.";
                                }
                                leaf label-range-size{
                                        type uint8;
                                        description
                                  "BIER label range.";
```

```
                                        }
                                        description
                                                "IPv4 mapping entries.";
                                }
                                list ipv6 {
                                        key "bitstringlength label";
                                        leaf bitstringlength {
                                                type uint16;
                                                description
                                      "BIER bitstringlength.";
                                        }
                                        leaf label{
                                                type uint32;
                                                description
                                          "BIER label.";
                                        }
                                        leaf label-range-size{
                                                type uint8;
                                                description
                                          "BIER label range.";
                                        }
                                        description
                                                "IPv6 mapping entries.";
                                }

                        description
                                           "Parameters associated with bier sub-domain."
;
                                }
                        description
                           "Parameters associated with bier.";
                        }
                        container birts{
                                list birt{
                                        key "sub-domain-id bitstringlength si";
                                        leaf sub-domain-id{
                                                type uint16;
                                                description
                                                        "BIER sub domain ID";
                                        }
                                        leaf bitstringlength{
                                                type uint16;
                                                description
                                          "BIER bitstringlength.";
                                        }
                                        leaf si{
                                                type uint16;
                                                description
                                          "BIER SI.";
```

```
                                }
                                leaf f-bm{
                                        type uint16;
                                        description
                                    "BIER Forwarding Bit Mask.";
                                }
                                leaf bier-mpls-in-label{
                                        type uint32;
                    description
                                        "BIER in-label.";
                                }
                                leaf bfr-nbr{
                                        type union{
                                        type inet:ipv4-address;
                    type inet:ipv6-address;
                                        }
                                        description
                                   "BIER BFR Neighbors.";
                                }
                                leaf bier-mpls-out-label{
                                        type uint32;
                     description
                                           "BIER out-label.";
                                }
                            description
                               "Shows Bit Index Routing Table.";
                            }
                 description
                    "BIER birt operational states.";
                }
        }

          /* Notifications */
    notification bfr-id-collision{
                leaf bfr-id{
                        type uint16;
                        description
                                "BIER BFR ID.";
        }
                description
                        "BFR ID received in the controlplane that caused BFR ID
                        collision.";
    }

    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 uint16;
                        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 domain ID collision";
    }

 }
```

8. Security Considerations

TBD.

9. Acknowledgements

TBD.

10. IANA Considerations

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

11. Normative 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-isis-extensions]
Ginsberg, L., Aldrin, S., Zhang, J., and T. Przygienda, "BIER support via ISIS", draft-ietf-bier-isis-extensions-00 (work in progress), April 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-02 (work in progress), August 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-00 (work in progress), April 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>.

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: April 20, 2016

T. Eckert
Cisco Systems, Inc.
G. Cauchie
Bouygues Telecom
October 18, 2015

Traffic Enginering for Bit Index Explicit Replication BIER-TE
draft-eckert-bier-te-arch-02

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 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 bierstrings 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 appropraitely agreed upon between all BFIR/BFER.

Currently, this specification has no considerations for BIER sub-domains.

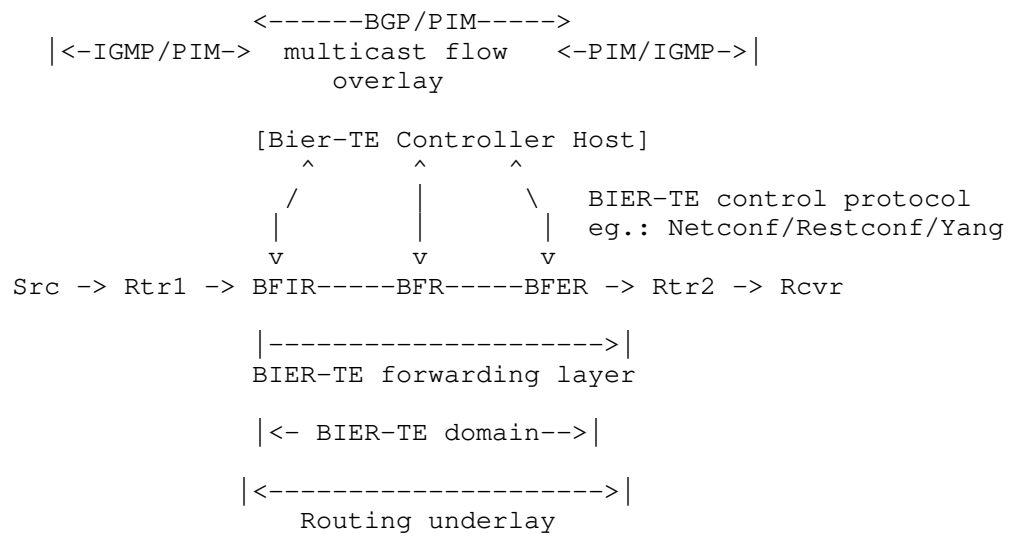
1.2. Requirements Language

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 informations 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 posible 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 FR 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 notification uses the BIER-TE FRR Adjacency Table to modify 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) 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 anbd 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 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 id desirable to support both BIER and BIER-TE forwarding in the same subdomain, then additional label would need to be assigned for BIER-TE forwarding.

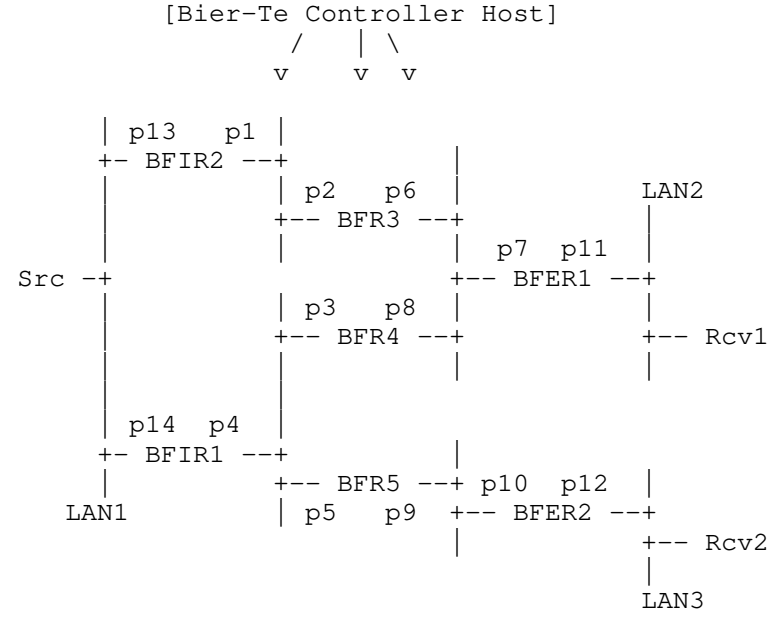
"forward_routed" requires an encapsulation permitting to unicast BIER-TE packets to a specific interface address on a target BFR. With MPLS encapsulation, this can simply be done via a label stack with that addresses label as the top label - followed by the label assigned to (SI,subdomain) - and if necessary (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

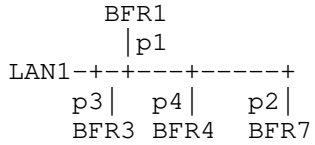
Leaf BFERs are BFERs where incoming BIER-TE packets never need to be forwarded to another BFR but are only sent to the BFER to exit the BIER-TE domain. For example, in networks where PEs are spokes connected to P routers, those PEs are Leaf BFIRs unless there is a U-turn between two PEs.

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.

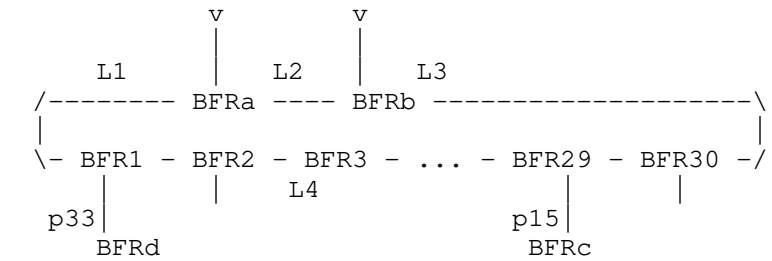
4.6. Rings

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 counterlockwise.

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

bit for which the replication is done. Likewise for the clockwise BitPosition for the counterlockwise copy. In result, the ring ingres 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

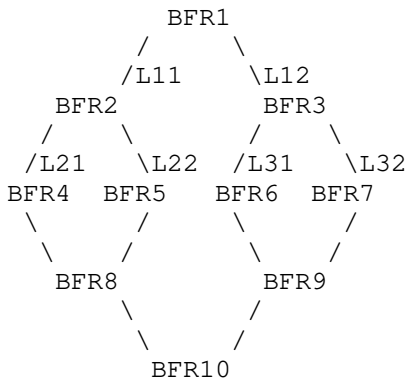
BIFT entry in BFR1:

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.

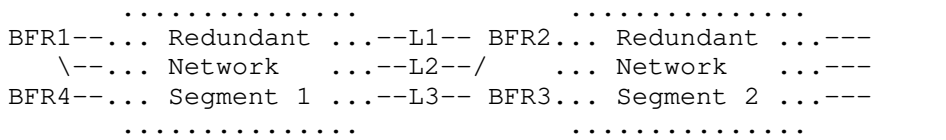
To resolve this issue, the ECMP adjaceny on BFR1 simply needs to be set up with a different seed than the ECMP adjacncies on BFR2/BFR3

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.

6.1. The BIER-TE Adjacency FRR Table (BTAFT)

The BIER-TE IF FRR Table exists in every BFR that is supporting BIER-TE FRR procedures. It is indexed by FRR Adjacency Index. Associated with each FRR Adjacency Index is a ResetBitmask, AddBitmask and BitPosition.

FRR Adjacency Index	BitPosition	ResetBitmask	AddBitmask
0:1	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

The BitPosition is the BP in the BIFT in which the FRR Adjacency is used

6.2. FRR in BIER-TE forwarding

The BIER-TE forwarding plane receives fast Up/Down notifications with the FRR Adjacency Index. From the 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 BPs to which it would forward. If it does, it will remove the ResetBitmask bits from the packets BitString and add the AddBitmask bits to the packets BitString.

Afterwards, normal BIER-TE forwarding occurs, taking the modified BitString into account.

6.3. 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 host has to determine whether a failure of the adjacency should be taken to indicate link or node failure. This is a policy decision.

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

3. 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.

4. 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.

5. 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.

6.4. 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.

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 AddtBitMaskByBT[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[BT]

    // 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 legth 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 choosen, 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 likey 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 efficieny 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 likelyhood, 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 detailled 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 braches": 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:

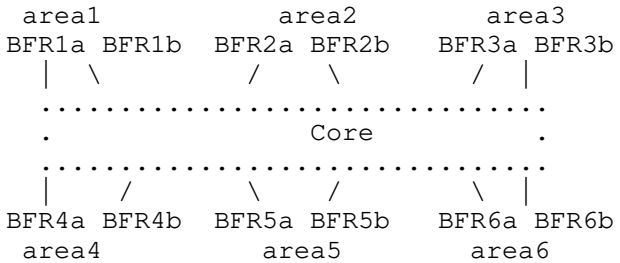
A. BIER and BIER-TE have different BFR-id in the same subdomain. This allows higher replication efficiency for BIER because their BFR-id can be assigned sequentially, while the bitstrings for BIER-TE will have also the additional bits for the topology. There is no relationship between a BFR BIER BFR-id 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 randomn 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 sufficienctly many spare bits for growths can help to aleviate 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 helpfull 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 intil 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, randomn 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, similarily 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 ingres a, bit ingres b, bit egres a, bit egres b. These bits will be used to pass BIER packets from any BFIR via any combination of ingres area

a/b BFR and egres 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 ingres area a or b border BFR and the egres area a or b border BFR, therefore allowing path engineering for those two "unicast" legs: 1) BFIR to ingres are edge and 2) core to egres area edge. Replication only happens inside the egres 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 mode-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 is not feasible or necessary, it is also possible for BIER-TE to leverage any existing form of "link" protection. For example: instead of dorectly 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.

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]

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., 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-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-02 (work in progress), August 2015.

[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

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: April 19, 2016

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

October 17, 2015

BIER support via ISIS
draft-ietf-bier-isis-extensions-01

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 April 19, 2016.

Copyright Notice

Table of Contents

1. Introduction

Bit Index Explicit Replication (BIER) [I-D.draft-ietf-bier-architecture-02] 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-02]. 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-02] 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).

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-TLVs 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

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 according BIER underlay.

Within such a domain, ISIS extensions are capable of carrying BIER information for multiple BIER sub-domains. Each sub-domain is uniquely identified by its subdomain-id and each subdomain can reside in any of the ISIS topologies [RFC5120]. The mapping of sub-domains to topologies is a local decision of each BFR currently but is advertised throughout the domain to ensure routing consistency.

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.

This RFC introduces a sub-TLV in the extended reachability TLVs to distribute such information about BIER sub-domains. To satisfy the requirements for BIER prefixes per [I-D.draft-ietf-bier-architecture-02] additional information will be carried in [I-D.draft-ietf-isis-prefix-attributes-01].

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 dos not have to be advertised by 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

All routers in the flooding scope of the BIER sub-TLVs MUST advertise a sub-domain within the same multi-topology. A router discovering a sub-domain advertised within a topology that is different from its own MUST report a misconfiguration of a specific sub-domain. Each router MUST compute BFTs for a sub-domain using only routers advertising it in the same multi-topology.

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>. Each router MUST compute BFTs for <MT,SD> using only routers having the same encapsulation as its own advertised encapsulation in BIER sub-TLV for <MT,SD>.

5.4. Tree Type

All routers in the flooding scope of the BIER TLVs MUST advertise the same tree type for a given <MT,SD>. In case of mismatch the behavior is analogous to Section 5.3.

5.5. Label Advertisements for MPLS encapsulated BIER sub-domains

Each router MAY advertise within the BIER MPLS Encapsulation sub-sub-TLV (Section 6.2) of a BIER Info sub-TLV (Section 6.1) for <MT,SD> (denoted as TLV<MT,SD>) for every supported bitstring length a valid starting label value and a non-zero range length. It MUST advertise at least one valid label value and a non-zero range length for the

required bitstring lengths per [I-D.draft-ietf-bier-architecture-02] in case it has computed itself as being on the BFT rooted at any of the BFRs with valid BFR-ids (except itself if it does NOT have a valid BFR-id) participating in <MT,SD>.

A router MAY decide to not advertise the BIER Info sub-TLV (Section 6.1) for <MT,SD> if it does not want to participate in the sub-domain due to resource constraints, label space optimization, administrative configuration or any other reasons.

5.5.1. Special Consideration

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-02]). Any router that violates this condition MUST be excluded from BIER BFTs for <MT,SD>.

5.6. BFR-id Advertisements

Each BFER MAY advertise with its TLV<MT,SD> the BFR-id that it has administratively chosen.

If a router discovers that two BFRs it can reach advertise the same value for BFR-id for <MT,SD>, it MUST report a misconfiguration and disregard those routers for all BIER calculations and procedures for <MT,SD> to align with [I-D.draft-ietf-bier-architecture-02]. It is worth observing that based on this procedure routers with colliding BFR-id assignments in <MT,SD> MAY still act as BFIRs in <MT,SD> but will be never able to receive traffic from other BFRs in <MT,SD>.

5.7. Flooding

BIER domain information SHOULD change and force flooding infrequently. Especially, the router SHOULD make every possible attempt to bundle all the changes necessary to sub-domains and ranges advertised with those into least possible updates.

6. Packet Formats

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

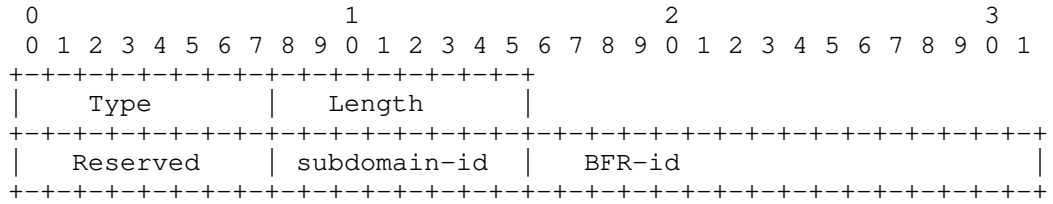
6.1. BIER Info sub-TLV

This sub-TLV carries the information for the BIER sub-domains that the router participates in as BFR. It can repeat multiple times for different multi-topology and sub-domain <MT,SD> combinations.

The sub-TLV carries a single <MT,SD> combination followed by optional sub-sub-TLVs specified within its context such as e.g. BIER MPLS Encapsulation per Section 6.2. If the same <MT,SD> combination is advertised more than once, only the first occurence of the sub-TLV MUST be used.

On violation of any of the following conditions, the receiving router SHOULD signal a misconfiguration condition. Further results are unspecified unless described in the according section of this RFC:

- The subdomain-id MUST be included only within a single topology.



Type: as indicated in IANA section.

Length: 1 octet.

Reserved: 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-02]. If set to the invalid BFR-id advertising router is not owning a BFR-id in the sub-domain.

6.2. BIER MPLS Encapsulation sub-sub-TLV

This sub-sub-TLV carries the information for the BIER MPLS encapsulation and the necessary label ranges per bitstring length for a certain <MT,SD> and is carried within the BIER Info sub-TLV (Section 6.1) that the router participates in as BFR.

On violation of any of the following conditions, the receiving router SHOULD signal a misconfiguration condition. Further results are by default unspecified unless explicitly described:

- The sub-sub-TLV MUST be included once AND ONLY once within the sub-TLV. If such a sub-sub-TLV is included more than once, only the first instance MUST be processed.

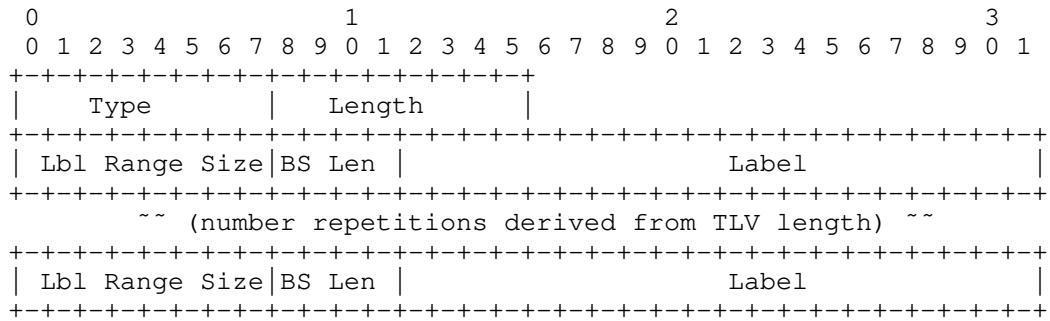
- Label ranges within the sub-sub-TLV MUST NOT overlap, otherwise the whole sub-sub-TLV MUST be disregarded and the violating routers are treated per further procedures in Section 5.3.

- Bitstring lengths within the sub-sub-TLV MUST NOT repeat, otherwise the whole sub-sub-TLV MUST be disregarded and the violating routers are treated per further procedures in Section 5.3.

- The sub-sub-TLV MUST include the required bitstring lengths encoded in precisely the same way as in [I-D.draft-ietf-bier-architecture-02].

- All label range sizes MUST be greater than 0.

- All labels MUST represent valid label values, otherwise the whole sub-sub-TLV MUST be disregarded and the violating routers are treated per further procedures in Section 5.3.



Type: value of 0 indicating MPLS encapsulation.

Length: 1 octet.

Local BitString Length (BS Len): Bitstring length for the label range that this router is advertising per [I-D.draft-ietf-bier-mpls-encapsulation-02]. 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. This MUST never be advertised as 0 (zero) and otherwise, this sub-sub-TLV must be treated as if not present for BFT calculations and a misconfiguration SHOULD be reported by the receiving router.

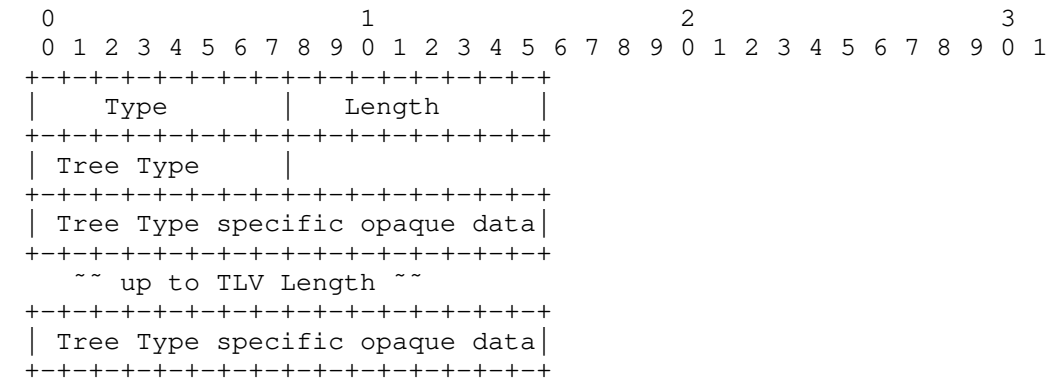
Label: First label of the range used on encapsulation for this BIER sub-domain for this bitstring length, 20 bits. The label is used for example by [I-D.draft-ietf-bier-mpls-encapsulation-02] to forward traffic to sets of BFERs.

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

This sub-sub-TLV carries the information of the 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). BIER implementation following this version of the RFC SHOULD NOT advertise this TLV.

On violation of any of the following conditions, the receiving router implementing this RFC SHOULD signal a misconfiguration condition. Further results are unspecified unless described further:

- The sub-sub-TLV MUST NOT be included more than once.
- The Tree Type MUST be 0 (SPF).



Type: value of 1 indicating BIER Tree Type.

Length: 1 octet.

Tree Type: The only supported value in this specification is 0 and indicates that BIER uses normal SPF computed reachability to construct BIFT. BIER implementation following this RFC MUST ignore the node for purposes of the sub-domain <MT,SD> if this field has any value except 0.

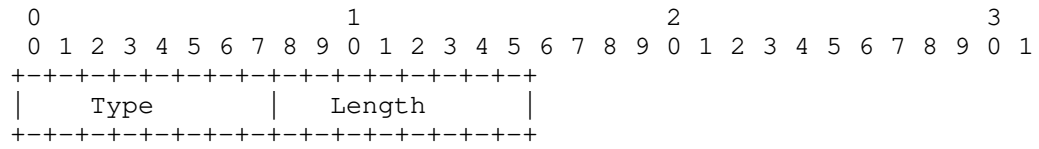
Tree type specific opaque data: Opaque data up to the length of the TLV carrying tree type specific parameters. For Tree Type 0 (SPF) no such data is included and therefore TLV Length is 1.

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.

On violation of any of the following conditions, the receiving router implementing this RFC SHOULD signal a misconfiguration condition. Further results are unspecified unless described further:

- The sub-sub-TLV MUST NOT be included more than once.



Type: value of 2 indicating BIER BSL conversion.

Length: 1 octet.

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-00] 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-02]
Wijnands et al., IJ., "Stateless Multicast using Bit Index Explicit Replication Architecture", internet-draft draft-ietf-bier-architecture-02.txt, July 2015.

[I-D.draft-ietf-bier-mpls-encapsulation-02]
Wijnands et al., IJ., "Bit Index Explicit Replication using MPLS encapsulation", internet-draft draft-ietf-bier-mpls-encapsulation-02.txt, Aug 2015.

[I-D.draft-ietf-bier-ospf-bier-extensions-00]
Psenak et al., P., "OSPF Extension for Bit Index Explicit Replication", internet-draft draft-ietf-bier-ospf-bier-extensions-00.txt, October 2014.

[I-D.draft-ietf-isis-prefix-attributes-01]
Ginsberg et al., U., "IS-IS Prefix Attributes for Extended IP and IPv6 Reachability", internet-draft draft-ietf-isis-prefix-attributes-01.txt, June 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>.

[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>.

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

BIER Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 10, 2016

G. Mirsky
T. Przygienda
Ericsson
A. Dolganow
Alcatel-Lucent

October 8, 2015

Path Maximum Transmission Unit Discovery (PMTUD) for Bit Index Explicit Replication (BIER) Layer
draft-mirsky-bier-path-mtu-discovery-00

Abstract

This document describes Path Maximum Transmission Unit Discovery (PMTUD) in Bit Indexed Explicit Replication (BIER) layer.

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 April 10, 2016.

Table of Contents

1. Introduction

In packet switched networks when a host seeks to transmit a sizable amount of data to a target destination the data is transmitted as a set of datagrams. In most cases it is more efficient to use the largest possible datagrams but so that these datagrams do not have to be fragmented at any point along the path from the host to the destination in order to avoid performance degradation caused by fragmentation. Fragmentation occurs on hops along the route where an Maximum Transmission Unit (MTU) is smaller than the size of the datagram. To avoid such fragmentation the MTU for each hop along a path from a host to a destination must be known to select an appropriate datagram size. Such MTU determination along a specific path is referred to as path MTU discovery (PMTUD).

[I-D.ietf-bier-architecture] introduces and explains Bit Index Explicit Replication (BIER) architecture and how it supports forwarding of multicast data packets. A BIER domain consists of Bit-Forwarding Routers (BFRs) that are uniquely identified by their respective BFR-ids. An ingress border router (acting as a Bit Forwarding Ingress Router (BFIR)) inserts a Forwarding Bit Mask (F-BM) into a packet. Each targeted egress node (referred to as a Bit Forwarding Egress Router (BFER)) is represented by Bit Mask Position (BMP) in the BMS. A transit or intermediate BIER node, referred as BFR, forwards BIER encapsulated packets to BFERs, identified by respective BMPs, according to a Bit Index Forwarding Table (BIFT).

1.1. Conventions used in this document

1.1.1. Terminology

BFR: Bit-Forwarding Router

BFER: Bit-Forwarding Egress Router

BFIR: Bit-Forwarding Ingress Router

BIER: Bit Index Explicit Replication

BIFT: Bit Index Forwarding Tree

F-BM: Forwarding Bit Mask

MTU: Maximum Transmission Unit

OAM: Operations, Administration and Maintenance

PMTUD: Path MTU Discovery

1.1.2. Requirements Language

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

2. Problem Statement

[I-D.ietf-bier-oam-requirements] sets forth the requirement to define PMTUD protocol for BIER domain. This document describes the extension to [I-D.kumarzheng-bier-ping] for use in BIER PMTUD solution.

Current PMTUD mechanisms [RFC1191], [RFC1981], and [RFC4821] are primarily targeted to work on point-to-point, i.e. unicast paths. These mechanisms use packet fragmentation control by disabling fragmentation of the probe packet. As result, a transient node that cannot forward a probe packet that is bigger than its link MTU sends to the ingress node an error notification, otherwise the egress responds with a positive acknowledgement. Thus, through series of iterations, decreasing and increasing size of the probe packet, the ingress node discovers the MTU of the particular path.

Thus applied such existing PMTUD solutions are inefficient for point-to-multipoint paths constructed for multicast traffic. Probe packets

must be flooded through the whole set of multicast distribution paths over and over again until the very last egress responds with a positive acknowledgement. Consider without loss of generality an example multicast network presented in Figure 1, where MTU on all links but one (B,D) is the same. If MTU on link (B,D) is smaller than the MTU on the other links, using existing PMTUD mechanism probes will unnecessary flood to leaf nodes E, F, and G for the second and consecutive times and positive responses will be generated and received by root A repeatedly.

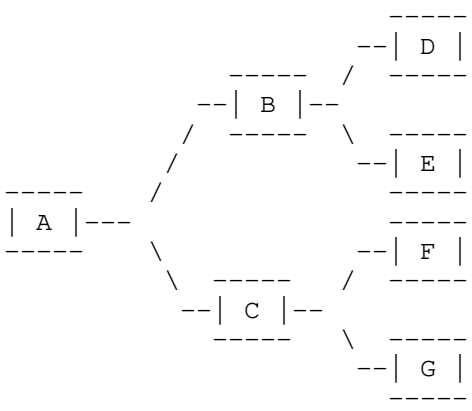


Figure 1: Multicast network

3. PMTUD Mechanism for BIER

A BFIR selects a set of BFERs for the specific multicast distribution. Such BFIR determines, by explicitly controlling subset of targeted BFERs and transmitting series of probe packets, the MTU of that multicast distribution path. The critical step is that in case of failure at an intermediate BFR to forward towards the subset of targeted downstream BFERs, the BFR responds with a partial (compared to the one it received in the request) bitmask towards the originating BFIR in error notification. That allows for retransmission of the next probe with smaller MTU address only towards the failed downstream BFERs instead of all BFERs addressed in the previous probe. In the scenario discussed in Section 2 the second and all following (if needed) probes will be sent only to the node D since MTU discovery of E, F, and G has been completed already by the first probe successfully.

[I-D.kumarzheng-bier-ping] introduced BIER Ping as transport-independent OAM mechanism to detect and localize failures in BIER

data plane. This document specifies how BIER Ping can be used to perform efficient PMTUD in BIER domain.

Consider network displayed in Figure 1 to be presentation of a BIER domain and all nodes to be BFRs. To discover MTU over BIER domain to BFERs D, F, E, and G BFIR A will use BIER Ping with Data TLV, defined in Section 3.1. Size of the first probe set to _M_max_ determined as minimal MTU value of BFIR's links to BIER domain. As been assumed in Section 2, MTUs of all links but link (B,D) are the same. Thus BFERs E. F, and G would receive BIER Echo Request and will send their respective replies to BFIR A. BFR B may pass the packet which is too large to forward over egress link (B, D) to the appropriate network layer for error processing where it would be recognized as BIER Echo Request packet. BFR B MUST send BIER Echo Reply to BFIR A and MUST include Downstream Mapping TLV, defined in [I-D.kumarzheng-bier-ping] setting its fields in the following fashion:

- MTU SHOULD be set to minimal MTU value among all egress BIER links that could be used to reach B's downstream BFERs;
- Address Type MUST be set to 0 [Ed.note: we need to define 0 as valid value for the Address Type field with the specific semantics to "Ignore" it.]
- I flag MUST be cleared;
- Downstream Interface Address field (4 octets) MUST be zeroed and MUST include in Egress Bitstring sub-TLV the list of all BFERs that cannot be reached because the attempted MTU turned out to be too small.

The BFIR will receive either of the two types of packets:

- a positive Echo Reply from one of BFERs to which the probe has been sent. In such case the bit corresponding to the BFER MUST be cleared from the BMS;
- a negative Echo Reply with bit string listing unreached BFERs and recommended MTU value MTU". The BFIR MUST add the bit string to its BMS and set size of the next probe as min(MTU, MTU")

If upon expiration of the Echo Request timer BFIR didn't receive any Echo Replies, then the size of the probe SHOULD be decreased. There are scenarios when an implementation of the PMTUD would not decrease the size of the probe. For example, if upon expiration of the Echo Request timer BFIR didn't receive any Echo Reply, then BFIR MAY continue to retransmit the probe using the initial size and MAY apply probe delay retransmission procedures. The algorithm used to delay

retransmission procedures on BFIR is outside the scope of this specification. The BFIR MUST continue sending probes using BMS until the bit string is clear or the discovery is declared unsuccessful. In case of convergence of the procedure, the size of the last probe indicates the MTU size that can be used for all BFERs in the initial BMS without incurring fragmentation.

Thus we conclude that in order to comply with the requirement in [I-D.ietf-bier-oam-requirements]:

- a BFR SHOULD support PMTUD;
- a BFR MAY use defined per BIER sub-domain MTU value as initial MTU value for discovery or use it as MTU for this BIER sub-domain to reach BFERs.

3.1. Data TLV for BIER Ping

There need to be control of probe size in order to support the BIER PMTUD. Data TLV format is presented in Figure 2.

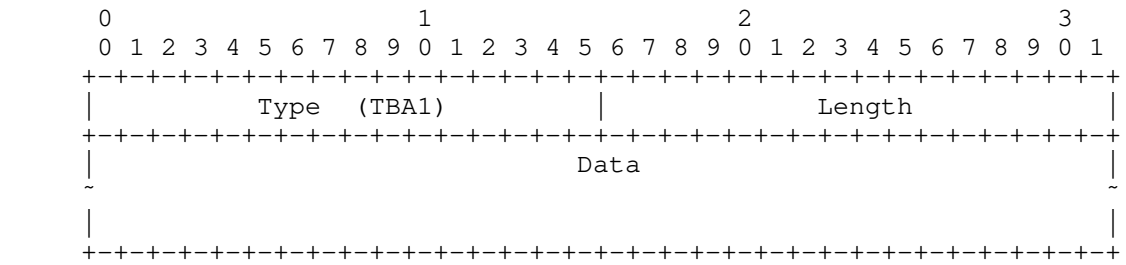


Figure 2: Data TLV format

- Type: indicates Data TLV, to be allocated by IANA Section 4.
- Length: the length of the Data field in octets.
- Data: n octets (n = Length) of arbitrary data. The receiver SHOULD ignore it.

4. IANA Considerations

IANA is requested to assign new Type value for Data TLV Type from its registry of TLV and sub-TLV Types of BIER Ping as follows:

Value	Description	Reference
TBA1	Data	This document

Table 1: Data TLV Type

5. Security Considerations

Routers that support PMTUD based on this document are subject to the same security considerations as defined in [I-D.kumarzheng-bier-ping]

6. Acknowledgement

TBD

7. References

7.1. Normative 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.kumarzheng-bier-ping]
Kumar, N., Pignataro, C., Akiya, N., Zheng, L., Chen, M., and G. Mirsky, "BIER Ping and Trace", draft-kumarzheng-bier-ping-01 (work in progress), July 2015.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <http://www.rfc-editor.org/info/rfc1191>.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <http://www.rfc-editor.org/info/rfc1981>.

[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>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <http://www.rfc-editor.org/info/rfc4821>.

7.2. Informative References

[I-D.ietf-bier-oam-requirements]
Mirsky, G., Nordmark, E., Pignataro, C., Kumar, N., Aldrin, S., Zheng, L., Chen, M., Akiya, N., and J. Networks, "Operations, Administration and Maintenance (OAM) Requirements for Bit Index Explicit Replication (BIER) Layer", draft-ietf-bier-oam-requirements-00 (work in progress), September 2015.

Authors' Addresses

Greg Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com

Tony Przygienda
Ericsson

Email: antoni.przygienda@ericsson.com

Andrew Dolganow
Alcatel-Lucent

Email: gandrew.dolganow@alcatel-lucent.com

BIER WG
Internet-Draft
Intended status: Standards Track
Expires: April 13, 2016

C. Wang
Z. Zhang
F. Hu
ZTE Corporation

October 11, 2015

BIER Use Case in VxLAN
draft-wang-bier-vxlan-use-case-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 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.

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.

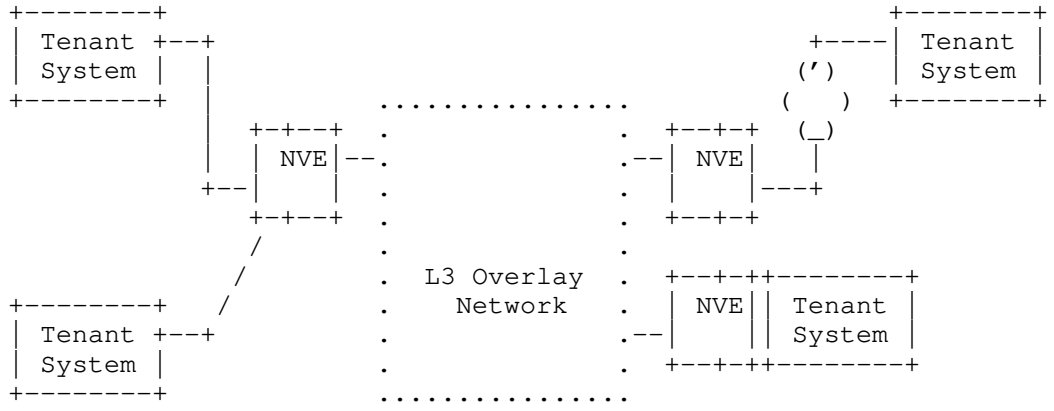


Figure 1: NVO3 Architecture

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 Virtual Network Instances (VNI) 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!_s 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 16 M 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 Virtual Network Instances (VNI) and the remote NVEs!_ addresses 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.

The following section tries to proposes how to take full advantage of BIER 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

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 NVE.

The principle of using BIER to forward BUM traffic is that: it requires each NVE to have a mapping between the VXLAN Virtual Network Instances (VNI) and the bit-string in which each bit represents exactly one remote NVE to forward the packet to. On other words, this requires the underlay network to support BIER which already be elaborated in [I-D.ietf-bier-architecture].

Already mentioned above, BIER requires no explicit tree-building protocols and maintains no multicast related per-flow state on the end nodes and intermediate nodes, just extends the IGP protocol or BGP protocol to advertise BIER-specific information to form BIER forwarding table in the BIER forwarding routers, such as NVEs and intermediate nodes in the data centers.

More importantly, as for how each NVE knows the other remote NVEs that belong to the same virtual network can also be discovered by additional BIER extensions. The following sections describe how to extend IGP protocol and BGP protocol to advertise VXLAN-specific information to tell each NVE where the other NVEs are in the same virtual network. As a result of this advertisement, each NVE creates the mapping between the VXLAN Virtual Network Instances (VNI) and the bit-string in which each bit represents exactly one remote NVE to forward the packet to.

4. BIER IS-IS extension for VXLAN-specific information

Specifically, in [I-D.ietf-bier-isis-extensions], there defines a new BIER Info sub-TLV which is illustrated in Figure 2. Here, extending a VXLAN-specific sub-sub-TLV to current BIER Info sub-TLV for IS-IS, a reference format is illustrated in Figure 3.

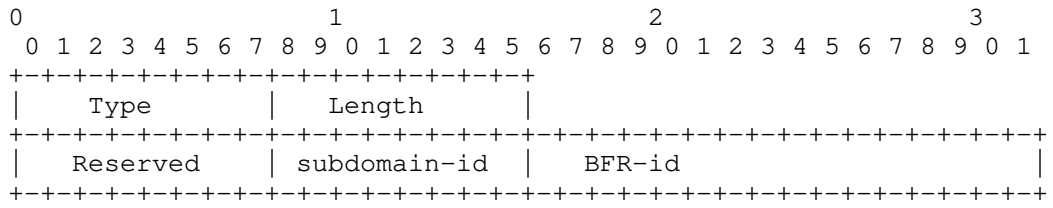


Figure 2: IS-IS BIER Info sub-TLV extensions for BIER-specific information

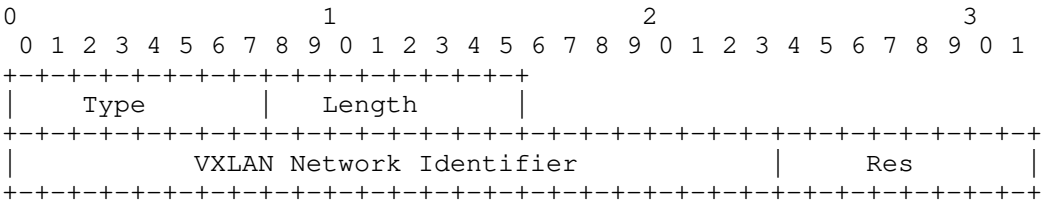


Figure 3: IS-IS VXLAN sub-sub-TLV extensions for VXLAN-specific information

Type:

indicates VXLAN sub-sub-TLV

Length: 1 cotet.

VXLAN Network Idenfifier:

indicates a virtual subnet

Then NVEs and intermediate nodes flood this VXLAN-specific sub-sub-TLV together with BIER Info sub-TLV through IS-IS in overlay network. When one NVE receives this IS-IS advertisement, this NVE builds a mapping between the receiving VNI in the VXLAN-specific sub-sub-TLV and the bit-string which represents the sending NVE and can extract

from the BIER Info sub-TLV. Once this NVE receives some other IS-IS advertisements which include the same VXLAN-specific sub-sub-TLV, it updates the bit-string in the mapping and adds the corresponding sending NVEs to the updated bit-string.

After finishing the above IS-IS flooding, each NVE knows where are the remote NVEs in the same virtual network. When receiving BUM traffic from the attached tenant, each NVE knows exactly how to forward this traffic to.

This can be used in both IPv4 network and IPv6 network.

5. BIER OSPF extension for VXLAN-specific information

Specifically, in [I-D.ietf-bier-ospf-bier-extensions], there defines a new BIER Info sub-TLV as well. Here, extending a VXLAN-specific sub-sub-TLV to current BIER Info sub-TLV for OSPF, a reference format is also illustrated in Figure 3.

Then NVEs and intermediate nodes flood this VXLAN-specific sub-sub-TLV together with BIER Info sub-TLV through OSPF in overlay network. When one NVE receives this OSPF advertisement, this NVE builds a mapping between the receiving VNI in the VXLAN-specific sub-sub-TLV and the bit-string which represents the sending NVE and can extract from the BIER Info sub-TLV. Once this NVE receives some other OSPF advertisements which include the same VXLAN-specific sub-sub-TLV, it updates the bit-string in the mapping and adds the corresponding sending NVEs to the updated bit-string.

After finishing the above OSPF flooding, each NVE knows where are the remote NVEs in the same virtual network. When receiving BUM traffic from the attached tenant, each NVE knows exactly how to forward this traffic to.

This can be used in both IPv4 network and IPv6 network.

6. BIER BGP extension for VXLAN-specific information

Specifically, in [I-D.ietf-bier-idr-extensions], there defines a new BGP path attribute referred to as the BIER attribute. Here, extending a VXLAN-specific sub-TLV to current BIER attribute TLV for BGP, a reference format is also illustrated in Figure 3.

Then NVEs and intermediate nodes flood this VXLAN-specific sub-TLV together with BIER attribute TLV through BGP in overlay network. When one NVE receives this BGP attribute, this NVE builds a mapping between the receiving VNI in the VXLAN-specific sub-TLV and the bit-string which represents the sending NVE and can extract from the BIER attribute TLV. Once this NVE receives some other BIER attribute TLV which include the same VXLAN-specific sub-TLV, it updates the bit-string in the mapping and adds the corresponding sending NVEs to the updated bit-string.

After finishing the above BGP advertisement, each NVE knows where are the remote NVEs in the same virtual network. When receiving BUM traffic from the attached tenant, each NVE knows exactly how to forward this traffic to.

This can be used in both IPv4 network and IPv6 network.

7. Considerations on BIER in data centers

TBD

8. Security Considerations

It will be considered in a future revision.

9. IANA Considerations

There need a new Type for VXLAN sub-sub-TLV.

10. References

10.1. 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>.

10.2. 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., Aldrin, S., Zhang, J., and T. Przygienda,

"BIER support via ISIS",
draft-ietf-bier-isis-extensions-00 (work in progress),
April 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-00 (work
in progress), April 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 Wang
ZTE Corporation
No.50 Software Avenue, Yuhuatai District
Nanjing
China

Email: wang.cui1@zte.com.cn

Zheng 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 Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

IJ. Wijnands
P. Pfister
Cisco Systems
October 19, 2015

Generic Multicast Router Election on LAN's
draft-wijnands-bier-mld-lan-election-00.txt

Abstract

When a host is connected to multiple multicast capable routers, each of these routers is a candidate to process the multicast flow for that LAN, but only one router should be elected to process it. This document proposes a generic multicast router election mechanism using Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) that can be used by any Multicast Overlay Signalling Protocol (MOSP). Having such generic election mechanism removes a dependency on Protocol Independent Multicast (PIM).

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 April 21, 2016.

Table of Contents

1. Introduction

Hosts connected to Local Area Networks (LAN) use Internet Group Management Protocol (IGMP) [RFC4605] or Multicast Listener Discovery (MLD) [RFC3810] to report their interest in a particular multicast flow. A multicast flow is identified by a Group or a combination of Group and Source address. Routers connected to a LAN listen to these membership reports and signal that information to the Multicast Overlay Signalling Protocol (MOSP). When a host is connected to multiple routers, each of these routers is a candidate to forward the multicast flow onto that LAN, but only one of them should forward the packets for a given flow to avoid duplication of Multicast packets. A similar requirement exists for hosts that are sending multicast traffic and are connected to multiple routers on a LAN. If multiple routers accept the multicast packets from the LAN, duplication may occur and/or routing loops may be created.

Protocol Independent Multicast (PIM) [RFC4601] is a MOSP and has a built-in mechanism to elect a Designated Router (DR) on the receiver LAN and a Designated Forwarder (DF) on the senders LAN. The DR/DF election avoids duplication and looping of multicast packets. Other existing or candidate MOSPs, like Border Gateway Protocol (BGP) [RFC6514], Multi-point Label Distribution Protocols (mLDP) [RFC6826], Locator ID Seperation Protocol (LISP) [RFC6830] and IGMP/MLD [I-D.pfister-bier-mld] have no embedded LAN DR/DF election mechanism. These MOSPs still rely on PIM to perform DR/DF election on LANs.

With the introduction of mLDP and Bit Indexed Explicit Replication (BIER) [I-D.ietf-bier-architecture], there is no dependency on PIM to

transport multicast packets through the network. Having a dependency on PIM just for DR/DF election is undesirable if PIM is not selected as the MOSP. This document proposes a generic DR/DF election which can be used by any MOSP without having a dependency on PIM. It potentially allows for different MOSPs to coexistence on single LANs.

2. Terminology and Definitions

Readers of this document are assumed to be familiar with the terminology and concepts of the documents listed as Normative References. For convenience, some of the more frequently used terms appear below.

LAN:
Local Area Network.

IGMP:
Internet Group Management Protocol.

MLD:
Multicast Listener Discovery.

mLDP:
Multipoint LDP.

PIM:
Protocol Independent Multicast.

ASM:
Any Source Multicast.

RP:
The PIM Rendezvous Point.

LISP:
Locator ID Seperation Protocol.

BIER:
Bit Indexed Explicit Replication.

MOSP:
Multicast Overlay Signalling Protocol. This is a protocol that is (potentially) capable of announcing multicast flow membership across the network between multicast routers. For example PIM, mLDP, BGP, IGMP, MLD and LISP.

3. Specification of Requirements

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].

4. Problem Statement

In the following sections we describe the requirements for DR/DF election in more detail for hosts that are multicast senders and receivers connected to multiple routers on a single LAN.

4.1. Receiver side

Consider the network below in Topology1.

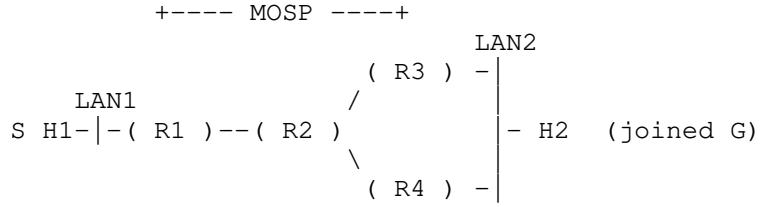


Figure 1

Suppose that H2 on LAN2 is joining a multicast Group G. The MOSP runs between R1, R3 and R4. Both R3 and R4 will receive the IGMP/MLD report, but only one of these should become the DR. One might consider that this problem can be detected and resolved by the MOSP. The MOSP could be enhanced to allow R1 to detect that both R2 and R4 are connected to the same LAN, and select only to forward the multicast flow to R3. That would solve the problem in the above topology, but would fail in the topology below:

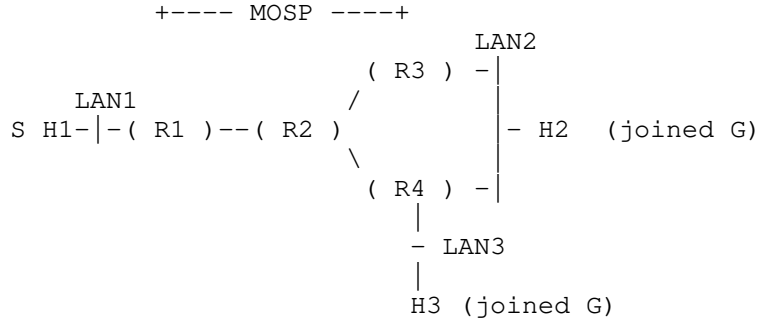


Figure 2

Consider that H3 on LAN3 joined the same multicast Group G. Since H3 is singly connected to R4, router R1 needs to forward the multicast flow to R4 in order for H3 to receive the packets. R4 does not have enough information to determine whether or not to forward on LAN2 for H2 when it receives the multicast packets due to H3. In other words, R4 needs DR state to avoid sending packets to H2 on LAN2.

4.2. Sender side

Consider the network below in Topology3.

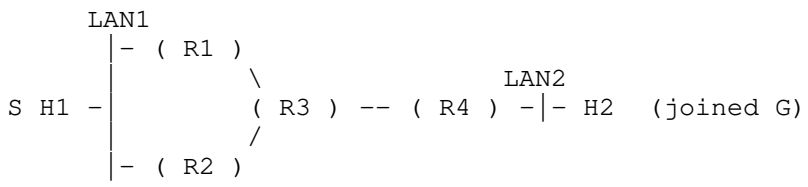


Figure 3

H1 is directly connected via a LAN1 to R1 and R2. H2 joins a multicast Group G, without specifying the Source. This is called Any Source Multicast (ASM). The MOSP signals R4s interest in Group G to R1 and R2. Note that there is no PIM deployed in this network and there is no Rendezvous Point (RP) that is a target for this receiving this Group membership. R4 has no information which routers in this network have multicast packets to sent for this Group. Since this is ASM, there may be multiple senders for this Group and H2 wants to receive them all. For that reason, R4 will use the MOSP to announce the membership to all edge nodes in the network (R1 and R2). This poses a potential problem since R1 and R2 are both directly connected to the Source S. If both R1 and R2 will forward the multicast packets to R4, H2 will receive duplicate packets. This is a problem that only occurs when a Source is dually connected to two or more routers connected to the BIER domain. This problem can be resolved by doing a Designated Forwarder (DF) election, similar to the DR election. If R1 and R2 are aware they are directly connected, an election will cause only one of them to forward the multicast packets into the BIER domain for a given (S,G) flow.

5. Proposal

As explain in Section 4, it is desirable to have a generic DR/DF election mechanism that can be used for existing and candidate MOSPs. Also, if a mix of MOSPs is used in the network, it is not obvious which MOSP is responsible for electing the DR/DF. If a single DR/DF is to be elected, and each MOSP does its own election, the MOSPs have

to negotiate among each other who will be responsible for DR/DF on a LAN. Independent of the MOSP, a single router connected to the LAN should be elected. It seems inefficient and unpractical to have each MOSP implement its own DR/DF election mechanism.

There is a process in the router that all the MOSPs depend on, that is the IGMP/MLD process. The DR/DF election is typically based on the Group address or Group and Source address of the multicast flow. This information is available in the IGMP/MLD process. In this document we propose to enhance the IGMP/MLD protocol to allow a DR/DF election among multicast routers connected to a LAN. As soon as a router is elected as DR/DF, it can select the MOSP that will be responsible to deliver the multicast flow to this router, and onwards onto the LAN(s).

IGMP/MLD has support for electing a Membership Querier based on the lowest IP address of the multicast routers sending out Membership Queries. It would be possible to use the elected Membership Querier as the DR/DF on a LAN. However, the authors believe that the Membership Querier procedures are not robust and extensible enough to be used DR/DF election on LANs. For example, if a new multicast router becomes active on a LAN, it will immediately assume the role of a Membership Querier, which can lead to duplication and/or looping of packets if also used as DR/DF. This duplication/looping will last until it learns about other Membership queriers with a lower IP address. Having two Membership queriers on the LAN has limited impact on the IGMP/MLD protocol it self, it would only cause more Membership Reports to be received.

The exact procedures to form a neighborship between IGMP/MLD routers will added in a later revision of this document.

6. Security Considerations

TBD.

7. IANA Considerations

TBD.

8. Acknowledgments

Many thanks to Neale Ranns and Greg Shepherd for their comments on this draft.

9. Normative 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.pfister-bier-mld]
Pfister, P., Wijnands, I., and M. Stenberg, "BIER Ingress Multicast Flow Overlay using Multicast Listener Discovery Protocols", draft-pfister-bier-mld-00 (work in progress), July 2015.

[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>.

[RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <http://www.rfc-editor.org/info/rfc3810>.

[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>.

[RFC4605] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, DOI 10.17487/RFC4605, August 2006, <http://www.rfc-editor.org/info/rfc4605>.

[RFC6514] Aggarwal, R., Rosen, E., Morin, T., and Y. Rekhter, "BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs", RFC 6514, DOI 10.17487/RFC6514, February 2012, <http://www.rfc-editor.org/info/rfc6514>.

[RFC6826] Wijnands, IJ., Ed., Eckert, T., Leymann, N., and M. Napierala, "Multipoint LDP In-Band Signaling for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6826, DOI 10.17487/RFC6826, January 2013, <http://www.rfc-editor.org/info/rfc6826>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <http://www.rfc-editor.org/info/rfc6830>.

Authors' Addresses

IJsbrand Wijnands
Cisco Systems
De Kleetlaan 6a
Diegem 1831
Belgium

Email: ice@cisco.com

Pierre Pfister
Cisco Systems
Paris
France

Email: pierre.pfister@darou.fr

BIER WG
Internet-Draft
Intended status: Standards Track
Expires: April 18, 2016

Sandy. Zhang
Bo. Wu
ZTE Corporation

October 16, 2015

Designed Routing in BIER Forwarding
draft-zhang-bier-designed-routing-00.txt

Abstract

BIER specifies a new architecture for the forwarding of multicast data packets. As the [I-D.ietf-bier-architecture] said, in the BIER domain, it does not require a protocol for explicitly building multicast distributing trees, nor does it require intermediate nodes to maintain any per-flow state. In some deployments, some specific multicast flows may be forwarded by special routing; it cannot be achieved by the forwarding rule that is provided. In this document we will defined a new routing list in BIER header, the packets will be steered according to the routing list.

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 April 18, 2016.

Table of Contents

1. Introduction

As the [I-D.ietf-bier-architecture] described, BIER specifies a new architecture for the forwarding of multicast data packets. And the extension of OSPF/ISIS is used to flood the BIER information to establish the BIER forwarding table. The calculation algorithm of IGP is calculating the shortest path to every BIER node. It means that the forwarding in BIER is along the shortest path.

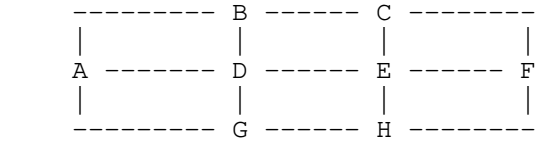


Figure 1: An example of packets forwarding

For example, in figure 1, in the forwarding table of node A, here supposes that the shortest path to node C is node B, the shortest path to node H is node G. If there is one specific multicast flow that should be forwarding to node C, node F and node H. Then the forwarding path will be node A --- node B --- node C, node A--- node D --- node E --- node F, node A --- node G --- node H. If we want to merge the flows in one optimizing path of node A --- node D --- node E --- node C/ node F/ node H, we cannot implement it due to the shortest forwarding algorithm.

This document specifies a way to forward multicast packets by designed routing. These multicast packets are encapsulated by specific header. The BIER nodes forward the packets by designed routing that is carried in specific BIER header. In figure 1, node A send the packets with designed routing list. The packets will be forwarded by the nodes list in the designed routing list of node A --- node D --- node E --- node C/ node F/ node H.

2. Designed Routing

The packets that are forwarded by BIER nodes are filled with designed routing in the BIER header, which carries the nodes list in order. The nodes are divided into differentiate levels that are classified by the distance from the ingress node.

2.1. The node's level

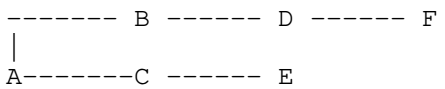


Figure 2

For example, in figure2, node A is ingress node, the node F and node E are egress nodes. Node B and C are the first level nodes that one specific multicast flow should be forwarded to, node D and E are the second level nodes that the flow should be forwarded to. There is only one node F in the third level.

3. Format of designed routing list

The designed routing list that is followed by the BIER header should be distinguished by some specific flag in the BIER header. One of the reserved bits may be used for the proposal. When the rightmost bit of the reserved field is set to 1, it indicates that the designed routing list is followed by the BIER header of the packet. As showed below:

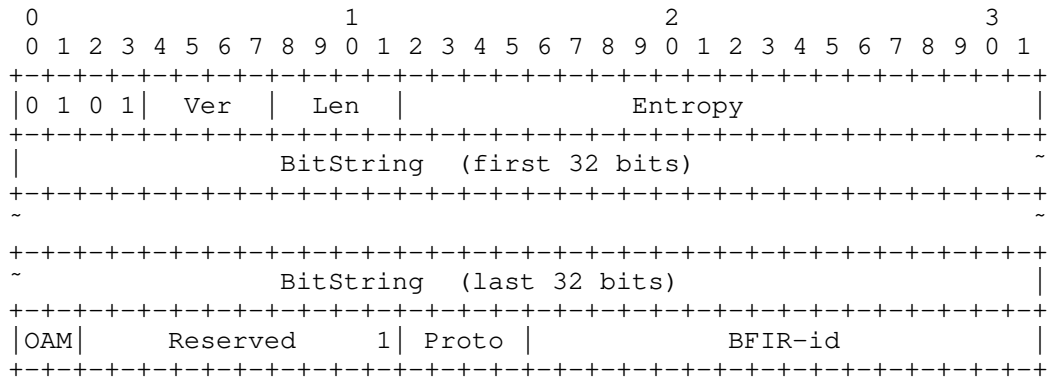


Figure 3

The designed routing list is comprised of a TLV, include type, length and value. The type of TLV indicates the routing information and form. There are many different types for the designed routing that may have many various formats, which may be expressed by bitstring, bfr-id list, or bfr-prefix list, and so on. The following section defines how to use the bitstring to express the designed routing list.

3.1. BitString

The BitPosition(BP) is used to indicate the nodes in BIER header. And it can be used in the list of designed routing. It's flexible to choose the corresponding BSL for packet's encapsulation. So it's also flexible to encapsulate the list of designed routing. The neighbor's format in the routing list should be used for two types.

3.1.1. unified BSL

Figure 4 illustrates using one unified BSL in the designed routing list, which means all level nodes in the list share the same BSL.

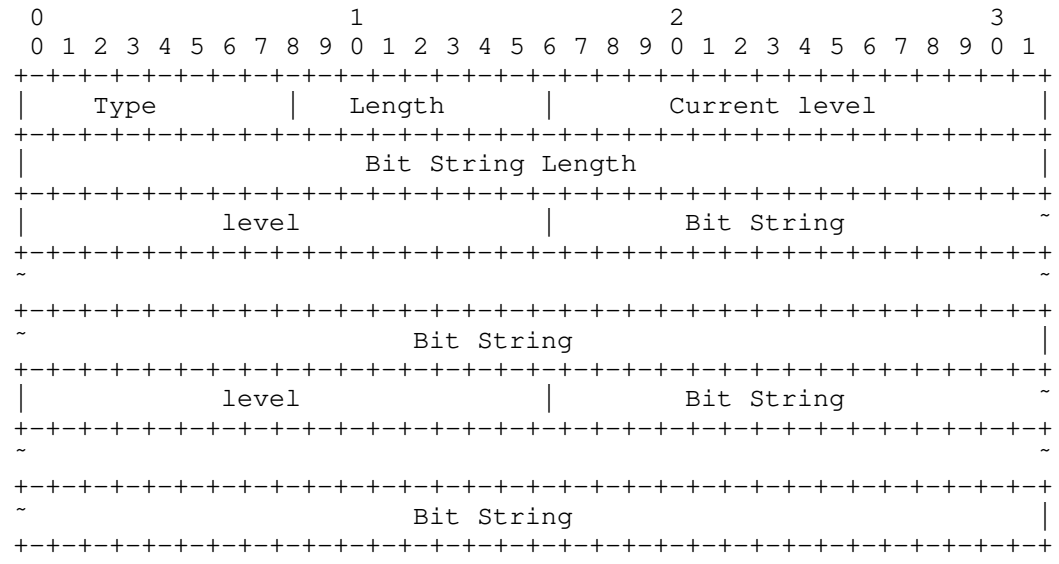


Figure 4

- Type : TBD. Suggest type 1.
- Length: The length of this TLV.
- Current level: Indicate the recent level that should be processed by the receiving node in the designed routing list. It will be increased by 1 after the processing of nodes in the designed routing.
- Bit String Length: Indicate the BSL that be used in the designed routing list.
- Level: Indicate the node's level of the designed routing.
- Bit String: All the nodes in the same level should be encapsulated in one bitstring.

3.1.2. Different BSL

Also different BSL should be used to express the nodes in the designed routing list. As sketched in Figure 5, different level nodes may be encapsulated in different BSL.

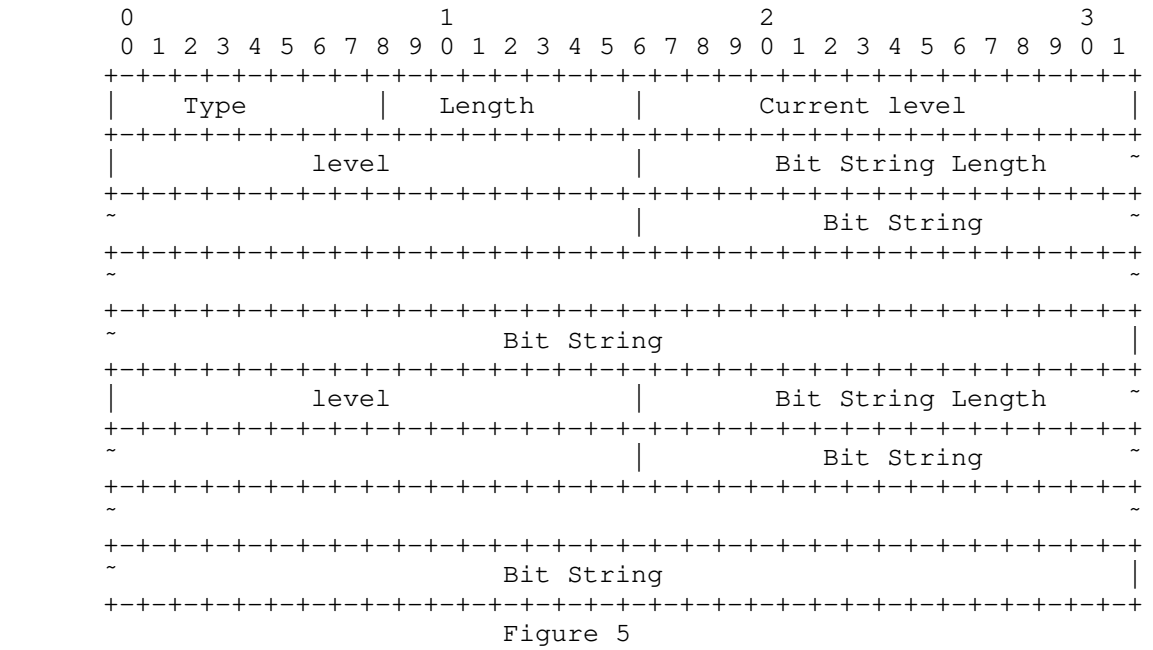


Figure 5

- Type : TBD. Suggest type 2.
- Length: The length of this TLV.
- Current level: Indicate the recent level that should be processed by the receiving node in the designed routing list. It will be increased by 1 after the processing of nodes in the designed routing.
- Level: Indicate the node's level of the designed routing.
- Bit String Length: Indicate the BSL that be used in this level.
- Bit String: All the nodes in the same level do be encapsulated in one bitstring.

4. Encapsulate the designed routing list

When the ingress node gathers all the egress nodes in the BIER domain, it should encapsulate the BIER header as regular function first when traffic arrives. If this traffic is classified to be forwarded by one designed routing, the ingress node then encapsulates the nodes' list in the designed routing list. The designed routing list may be provided from the controller or the module of PCE, and so on. The nodes in one same level should be encapsulated together.

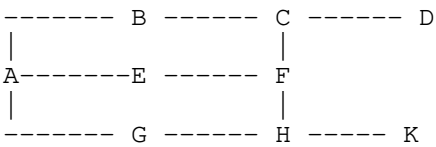


Figure 6

For example, there are several nodes in one BIER domain. Suppose that for one specific multicast flow, the ingress node is A, and the egress nodes are D, F and K. The designed routing list is A-E-F-C-D, A-E-F-H-K. What is more, node B is the shortest path next hop from A to D, node G is the shortest path next hop from A to K.

The designed routing is divided into four levels, the first level includes node E. The second level includes node F. The third level includes node C and node H. The fourth level includes node D and node K.

Node A encapsulates all the nodes in the path to the designed routing list. The current level is set to 1.

5. Decapsulate the packet

When the packet is forwarded to one of the egress nodes, the egress node will detect that it is one of the egress nodes due to the bitstring in the BIER header. The egress nodes decapsulate the BIER header and forward it to the previous domain by reductive format. And if the egress node notices that there are still many nodes in the designed routing list that should be forwarded, the egress node will forward packets to next node due to the designed routing list.

6. Forwarding treatment

When the BIER nodes receive a packet that is encapsulated by BIER header and designed routing list extension, the BIER nodes will process the designed routing list in the BIER header and forward the packet according to the list.

The current nodes receive a packet from the control plane or one previous node. The current nodes will process the designed routing list due to the indication in the BIER header. If there is no indication of designed routing list in the BIER header, the current nodes will forward the packet according to the BIER forwarding defined in [I-D.ietf-bier-architecture].

If there is a designed routing list in the BIER header, the current nodes will forward the packet due to the rules below.

At first, if the node is one of the egress nodes due to the BIER header, the nodes should decapsulate the packet and forward it to outside.

Second, if the node detect that there is a designed routing list in the BIER header, the node will process the designed routing list, and forward the packet to the nodes in the list. The detail of processing is described in the subsection.

At last, the node increases the current level in the designed routing list, and forwards the packet to the next level nodes in the list.

6.1. Process the designed routing list

Step 1, the node gets the nodes' list in the designed routing list belong to current level. For example, if the value of current level field is set to 1, then the node will get the nodes that are encapsulate in level 1.

Step 2, the node looks up the BIER forwarding table and gets the table items which the next-hop are same to the nodes that get from the designed routing list.

Step 3, the node copies the bitstring in the BIER header and updates it by AND'ing with all the table items. After INVERSE the result, do And'ing with the original bitstring. Then the result is reserved egress nodes. It can be simplified to Rev-enodes.

Step 4, the node update the origin bitstring by AND'ing with one item that are get from step2, and merge the result with the Rev-enodes, increase the current level field by 1 and forwards the packet to the next-hop of the item. If there are several items get from step2, the node will do the processing with all the items.

Every node in the designed routing list repeats the step, the packet will be forwarded by the nodes in the designed routing list. At last the packet is forwarded to outside of the BIER domain.

As the example of section 4, Node A receives the packet from the control plane. The current level field is 1. Node A gets node E from the designed routing list due to the current level. Then node A lookups the BIER forwarding table and gets the item which the next-hop is node E.

Because the shortest path from node A to node D and K is not go through node E, so the node A will get the Rev-enodes that include node D and K due to the steps in section 6.1. Node A forwards the

packet to node E, with the bitstring in the BIER header is set to D, F and K, and the current level field is set to 2.

Node E receives the packet from node A, finds the items from the BIER forwarding table which the next-hop is node F, do the same steps, set the current level field to 3, and then forwards the packet to F.

Node F receives the packet from node E. At first node F decapsulates the packet and forwards it outside of the BIER domain due to the bitstring in the BIER header. Then node F finds the items from the BIER forwarding table which the next-hop is node C and H, does the same steps in section 6.1, increase the current level value and forward to node C and H.

Node C and H receive the packet from node F separately, repeat the steps, increase the current level value to 4 and forwards it to node D and K.

Node D and node K receive the packet from node F, repeat the steps, decapsulate the packet and forward it out the BIER domain.

7. Security Considerations

For general BIER Security Considerations.

8. IANA Considerations

IANA is requested to allocate types in TLVs of BIER header.

9. Normative 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-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-02 (work in progress), August 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

Sandy Zhang
ZTE Corporation
No. 50 Software Ave, Yuhuatai Distinct
Nanjing 210000
China

Phone: +86-025-88014634
Email: zhang.zheng@zte.com.cn

Bo Wu
ZTE Corporation
No. 50 Software Ave, Yuhuatai Distinct
Nanjing 210000
China

Phone: +86-025-88016575
Email: wu.bo@zte.com.cn