Networking Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 24, 2017

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

July 23, 2016

BGP Link-State extensions for BIER
draft-chenvgovindan-bier-bgp-ls-bier-ext-01

Abstract

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

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 24, 2017.

Table of Contents

1. Introduction 2
2. Conventions used in this document 3
3. BGP-LS Extensions for BIER 3
3.1. The BIER TLV 3
3.1.1. The BIER MPLS Encapsulation Sub-TLV 4
3.2. The BIER-TE TLV 5
4. IANA Considerations 5
5. Security Considerations 6
6. Acknowledgements 6
7. Normative references 6
Authors' Addresses 7

1. Introduction

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

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

2. Conventions used in this document

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

3. BGP-LS Extensions for BIER

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

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

3.1. The BIER TLV

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

The following BIER TLV is defined:

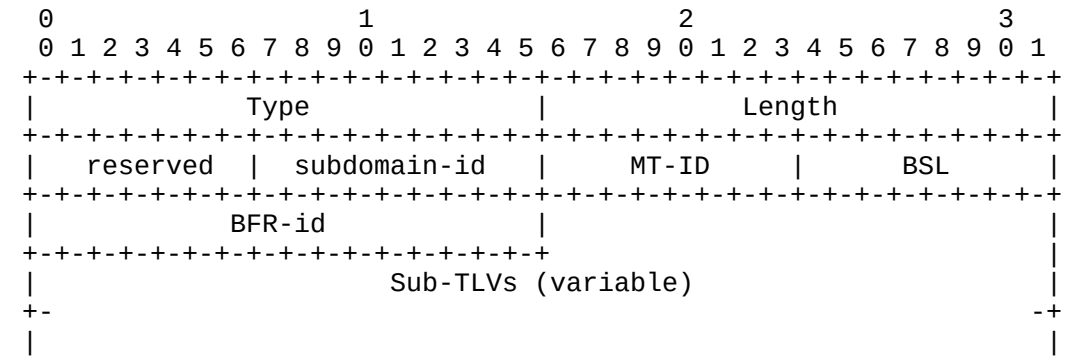


Figure 1

Type: TBD.

Length: 2 octet.

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

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

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

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

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

3.1.1. The BIER MPLS Encapsulation Sub-TLV

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

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

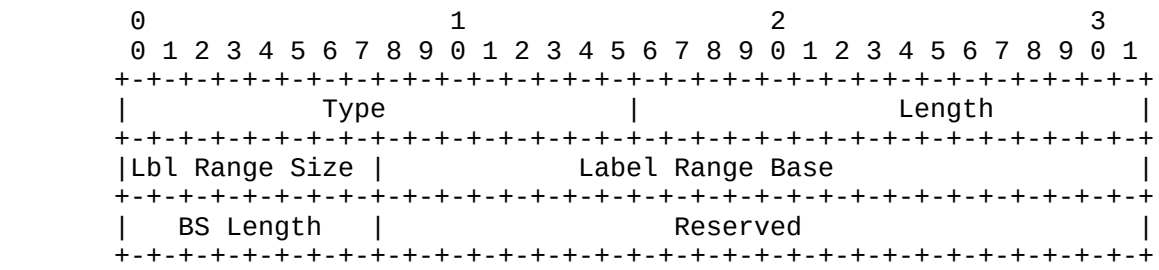


Figure 2

Type: TBD.

Length: 2 octet.

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

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

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

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

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

3.2. The BIER-TE TLV

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

The section will be added in next version.

4. IANA Considerations

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

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

Table 1: The new Prefix Attribute TLV

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

Table 2: The new Prefix Attribute Sub-TLV

5. Security Considerations

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

6. Acknowledgements

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

7. Normative references

[I-D.ietf-bier-architecture]
Wijnands, I., Rosen, E., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast using Bit Index Explicit Replication", draft-ietf-bier-architecture-04 (work in progress), July 2016.

[I-D.ietf-bier-isis-extensions]
Ginsberg, L., Przygienda, T., Aldrin, S., and Z. Zhang, "BIER support via ISIS", draft-ietf-bier-isis-extensions-02 (work in progress), March 2016.

[I-D.ietf-bier-mpls-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication in MPLS Networks", draft-ietf-bier-mpls-encapsulation-05 (work in progress), July 2016.

[I-D.ietf-bier-ospf-bier-extensions]
Psenak, P., Kumar, N., Wijnands, I., Dolganow, A., Przygienda, T., Zhang, Z., and S. Aldrin, "OSPF Extensions for BIER", draft-ietf-bier-ospf-bier-extensions-02 (work in progress), March 2016.

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

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

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

Authors' Addresses

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

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

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

Email: zhang.zheng@zte.com.cn

Vengada Prasad Govindan
Cisco

Email: venggovi@cisco.com

IJsbrand Wijnands
Cisco
De Kleetlaan 6a
Diegem 1831
Belgium

Email: ice@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 20, 2018

T. Eckert, Ed.
Huawei
G. Cauchie
Bouygues Telecom
W. Braun
M. Menth
University of Tuebingen
November 16, 2017

Traffic Engineering for Bit Index Explicit Replication BIER-TE
draft-eckert-bier-te-arch-06

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 3
1.1. Overview 3
1.2. Requirements Language 4
2. Layering 4
2.1. The Multicast Flow Overlay 5
2.2. The BIER-TE Controller Host 5
2.2.1. Assignment of BitPositions to adjacencies of the network topology 6
2.2.2. Changes in the network topology 6
2.2.3. Set up per-multicast flow BIER-TE state 6
2.2.4. Link/Node Failures and Recovery 6
2.3. The BIER-TE Forwarding Layer 7
2.4. The Routing Underlay 7
3. BIER-TE Forwarding 7
3.1. The Bit Index Forwarding Table (BIFT) 7
3.2. Adjacency Types 8
3.2.1. Forward Connected 8
3.2.2. Forward Routed 9
3.2.3. ECMP 9
3.2.4. Local Decap 9
3.3. Encapsulation considerations 10
3.4. Basic BIER-TE Forwarding Example 10
3.5. Forwarding comparison with BIER 12
4. BIER-TE Controller Host BitPosition Assignments 13
4.1. P2P Links 13
4.2. BFER 14
4.3. Leaf BFERs 14
4.4. LANs 14
4.5. Hub and Spoke 15
4.6. Rings 15
4.7. Equal Cost MultiPath (ECMP) 16

4.8. Routed adjacencies 18
4.8.1. Reducing BitPositions 18
4.8.2. Supporting nodes without BIER-TE 18
5. Avoiding loops and duplicates 18
5.1. Loops 18
5.2. Duplicates 19
6. BIER-TE Forwarding Pseudocode 19
7. Managing SI, subdomains and BFR-ids 20
7.1. Why SI and sub-domains 21
7.2. Bit assignment comparison BIER and BIER-TE 22
7.3. Using BFR-id with BIER-TE 22
7.4. Assigning BFR-ids for BIER-TE 23
7.5. Example bit allocations 24
7.5.1. With BIER 24
7.5.2. With BIER-TE 25
7.6. Summary 26
8. BIER-TE and Segment Routing 26
9. Security Considerations 27
10. IANA Considerations 27
11. Acknowledgements 27
12. Change log [RFC Editor: Please remove] 27
13. References 29
Authors' Addresses 29

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 incremental deployment and a Fast ReRoute (FRR) extension for link and node protection is given in [I-D.eckert-bier-te-frr]. Because BIER-TE like BIER operates without explicit in-network tree-building but also supports traffic engineering, it is more similar to Segment Routing (SR) than RSVP-TE.

The key differences over BIER are:

- BIER-TE replaces in-network autonomous path calculation by explicit paths calculated offpath by the BIER-TE controller host.

- In BIER-TE every BitPosition of the BitString of a BIER-TE packet indicates one or more adjacencies - instead of a BFER as in BIER.

- BIER-TE in each BFR has no routing table but only a BIER-TE Forwarding Table (BIFT) indexed by SI:BitPosition and populated with only those adjacencies to which the BFR should replicate packets to.

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

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

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

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

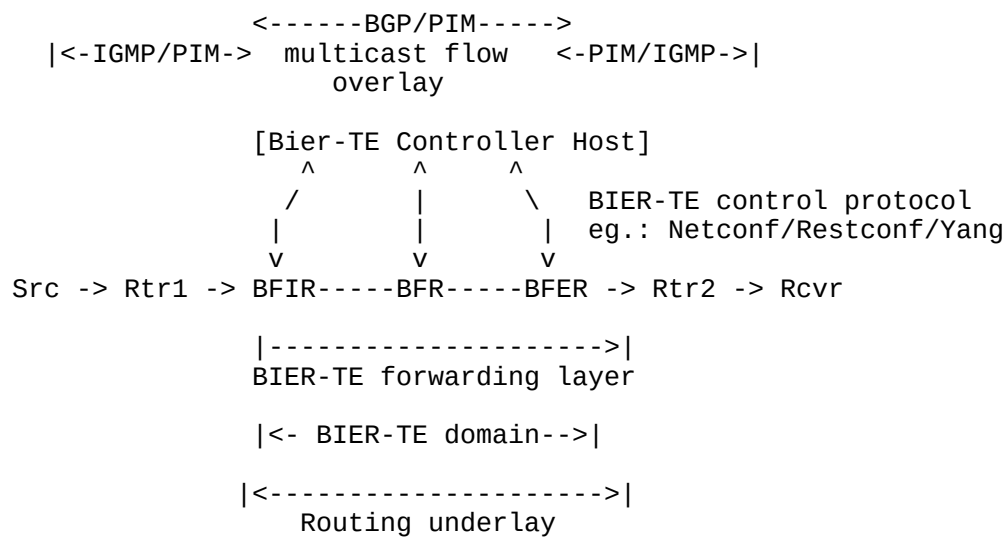
1.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Layering

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

Picture 2: Layers of BIER-TE



2.1. The Multicast Flow Overlay

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

2.2. The BIER-TE Controller Host

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

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

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

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

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

2.2.1. Assignment of BitPositions to adjacencies of the network topology

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

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

2.2.2. Changes in the network topology

If the network topology changes (not failure based) so that adjacencies that are assigned to BitPositions are no longer needed, the controller can re-use those BitPositions for new adjacencies. First, these BitPositions need to be removed from any BFIR flow state and BFR BIFT state, then they can be repopulated, first into BIFT and then into the 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 in [I-D.eckert-bier-te-frr]. It can also more slowly react by recalculating the BitStrings of affected multicast flows. This reaction is slower than the FRR procedure because the controller needs to receive link/node up/down indications, recalculate the desired BitStrings and push them down into the BFIRs. With FRR, this

is all performed locally on a BFR receiving the adjacency up/down notification.

2.3. The BIER-TE Forwarding Layer

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

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 7 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 and it introduces a "seed" parameter to allow engineering traffic to be polarized or randomized across multiple hops.

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

3.2.4. Local Decap

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

3.3. Encapsulation considerations

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

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

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

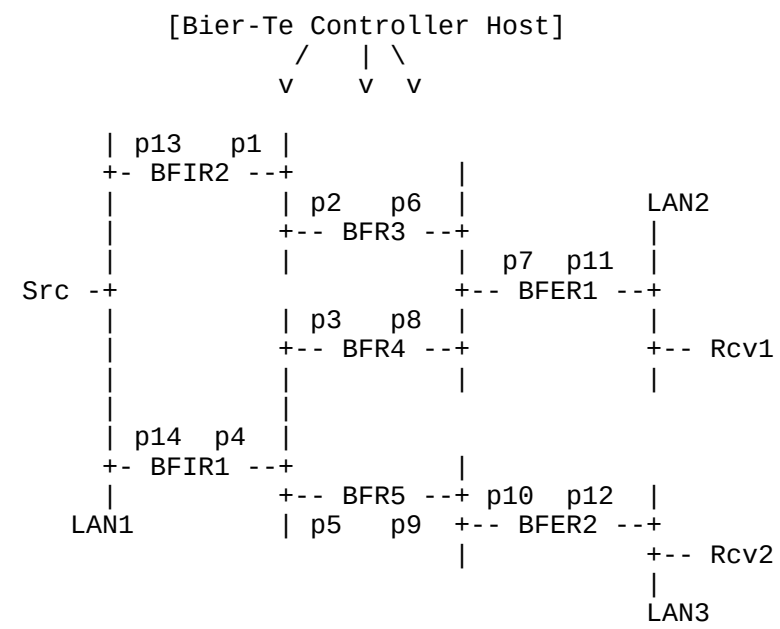
"forward_routed" requires an encapsulation permitting to unicast BIER-TE packets to a specific interface address on a target BFR. With MPLS encapsulation, this can simply be done via a label stack with that addresses label as the top label - followed by the label assigned to (SI,subdomain) - and if necessary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

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

3.4. Basic BIER-TE Forwarding Example

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

Picture 1: Forwarding Example



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

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

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

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

...and so on.

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

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

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

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

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

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

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

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

3.5. Forwarding comparison with BIER

Forwarding of BIER-TE is designed to allow common forwarding hardware with BIER. Like BIER, the core of BIER-TE forwarding are BIFTs with bitstring size number of entries: One for each bit of the bitstring in the processed packet (consider that 256 is the most common size).

When a packet is received, the BIFT to process needs to be selected. This is based on SI and subdomain like in BIER. How SI and subdomain are indicated is subject to the BIER-TE encapsulation, but not BIER-T itself. It is expected that the mechanisms for encapsulation will be very similar if not the same to BIER, but this is subject to followup work.

There are some key difference between the BIFT in BIER and BIER-TE:

In BIER-TE, each entry in the BIFT can have a list of 0 or more adjacencies. A separate copy of the packet is made for each adjacency. In BIER, each BIFT entry has at most one adjacency (BFR-

NBR). In BIER, different bits can not be processed independently directly: Only one packet copy is to be sent for all bits in the packet with the same adjacency, which is why the forwarding procedure specifies how to sequentially identify those bits and avoid duplication. In BIER-TE there are no mutual dependencies between bit adjacencies, so all bits of a BIER-TE bitstring could be procssed independently in parallel.

In BIER the BIFT has adjacencies for all BFR-ids assigned to BFER and reachable in the IGP. In BIER-TE the BIFT only has adjacencies for bits that are adjacent hops - intermediate or BFER. In forwarding, this can be treated via the same lookup logic except that in BIER-TE there is no step modifyin the original packet and the packet copy bitstring with the FBM. Instead, all the bits locally processed are reset in the original packet before looking up bits in the BIFT (˜MyBitsOfInterest). Only for an adjacency with the "DNR" (Do Not Reset) bit set would the bit in the bitstring not be set again as part of processing of the adjacency.

In summary, implementations of BIER forwarding that are to be extended to also support BIER-TE forwarding primarily need to consider how they can ensure that individual bit lookups can result in a sequence of more than one copy to be made (as opposed to one in BIER), and they need to see that they can accordingly reset bits in the bitstring differently for BIER (per-packet) vs. BIER-TE (per-paket-copy).

4. BIER-TE Controller Host BitPosition Assignments

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

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

4.1. P2P Links

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

4.2. BFER

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

4.3. Leaf BFERs

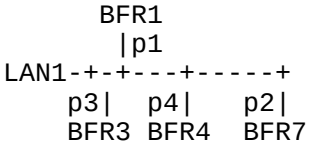
Leaf BFERs are BFERs where incoming BIER-TE packets never need to be forwarded to another BFR but are only sent to the BFER to exit the BIER-TE domain. For example, in networks where PEs are spokes connected to P routers, those PEs are Leaf BFIRs unless there is a U-turn between two PEs.

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

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

4.4. LANs

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



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

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

4.5. Hub and Spoke

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

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

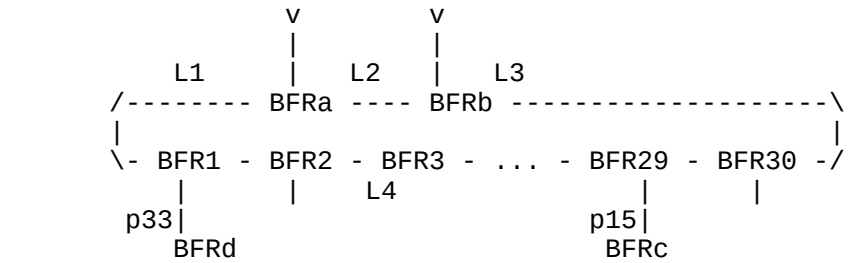
4.6. Rings

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

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

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

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



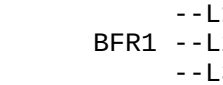
Note that this example only permits for packets to enter the ring at BFRa and BFRb, and that packets will always travel clockwise. If packets should be allowed to enter the ring at any ring BFR, then one would have to use two ring BitPositions. One for clockwise, one for counterclockwise.

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

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

4.7. Equal Cost MultiPath (ECMP)

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



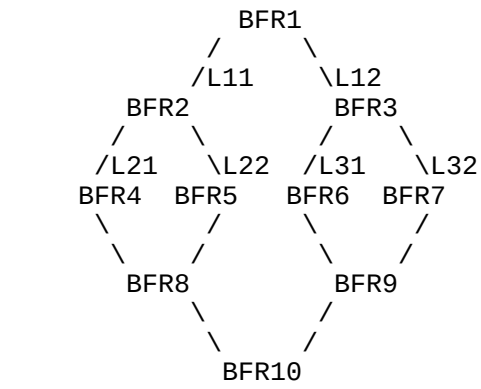
BIFT entry in BFR1:

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

BIFT entry in BFR2:

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

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



BIFT entry in BFR1:

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

BIFT entry in BFR2:

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

BIFT entry in BFR3:

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

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

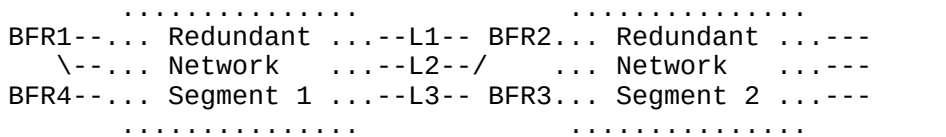
To resolve this issue, the ECMP adjacency on BFR1 simply needs to be set up with a different seed than the ECMP adjacencies on BFR2/BFR3

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

4.8. Routed adjacencies

4.8.1. Reducing BitPositions

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



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

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

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

4.8.2. Supporting nodes without BIER-TE

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

5. Avoiding loops and duplicates

5.1. Loops

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

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

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

5.2. Duplicates

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

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

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

6. BIER-TE 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 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

    // 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)
}
```

7. 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.

7.1. Why SI and sub-domains

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

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

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

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

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

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

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

7.2. Bit assignment comparison BIER and BIER-TE

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

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

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

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

7.3. Using BFR-id with BIER-TE

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

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

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

"Independent branches": For a given application or (set of) trees, the branches from a BFIR to every BFER are independent of the

branches to any other BFER. For example, shortest part trees have independent branches.

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

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

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

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

7.4. 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:

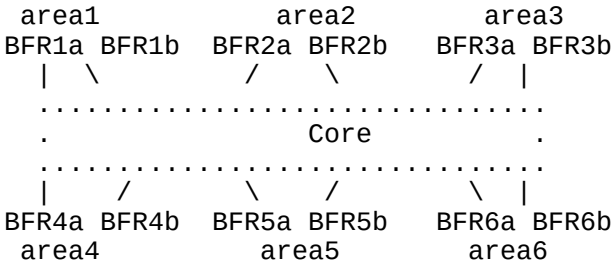
A. BIER and BIER-TE have different BFR-id in the same subdomain. This allows higher replication efficiency for BIER because their BFR-id can be assigned sequentially, while the bitstrings for BIER-TE will have also the additional bits for the topology. There is no relationship between a BFR BIER BFR-id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

7.5. Example bit allocations

7.5.1. With BIER

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



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

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

Given how networks can grow over time, replication efficiency in an area will also easily go down over time when BFR-id are network wide allocated sequentially over time. An area that initially only has

BFR-id in one SI might end up with many SI over a longer period of growth. Allocating SIs to areas with initially sufficiently many spare bits for growths can help to alleviate this issue. Or renumber BFR-id after network expansion. In this example one may consider to use 6 SI and assign one to each area.

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

7.5.2. With BIER-TE

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

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

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

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

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

In addition, we use 4 bits in each SI: bia, bib, bea, beb: bit ingress a, bit ingress b, bit egress a, bit egress b. These bits will be used to pass BIER packets from any BFIR via any combination of ingress area a/b BFR and egress area a/b BFR into a specific target area. These bits are then set up with the right forward_routed adjacencies on the BFIR and area edge BFR:

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

forward_routed(BFRka) and beb in BIFT:SI=k with forward_routed(BFRkb).

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

7.6. 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.

8. 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.

9. 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.

10. IANA Considerations

This document requests no action by IANA.

11. Acknowledgements

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

12. Change log [RFC Editor: Please remove]

draft-eckert-bier-te-arch:

00: Source now on http://www.github.com/toerless/bier-te-arch Please open issues on the github for change/improvement requests to the document - in addition to posting them on the list (bier@ietf.). Thanks!.

- Added overview of differences between BIER, BIER-TE forwarding.

draft-eckert-bier-te-arch:

06: Added forwarding comparison with BIER.

05: Author affiliation change only.

04: Added comparison to Live-Live and BFIR to FRR section (Eckert).

04: Removed FRR content into the new FRR draft [I-D.eckert-bier-te-frr] (Braun).

- Linked FRR information to new draft in Overview/Introduction

- Removed BTAFT/FRR from "Changes in the network topology"

- Linked new draft in "Link/Node Failures and Recovery"

- Removed FRR from "The BIER-TE Forwarding Layer"

- Moved FRR section to new draft

- Moved FRR parts of Pseudocode into new draft

- Left only non FRR parts

- removed FrrUpDown(..) and //FRR operations in ForwardBierTePacket(..)

- New draft contains FrrUpDown(..) and ForwardBierTePacket(Packet) from bier-arch-03

- Moved "BIER-TE and existing FRR to new draft

- Moved "BIER-TE and Segment Routing" section one level up

- Thus, removed "Further considerations" that only contained this section

- Added Changes for version 04

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

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

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

02: Added Section 7 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.

13. 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-08 (work in progress), September 2017.

[I-D.ietf-bier-mpls-encapsulation]
Wijnands, I., Rosen, E., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication in MPLS and non-MPLS Networks", draft-ietf-bier-mpls-encapsulation-12 (work in progress), October 2017.

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

Authors' Addresses

Toerless Eckert (editor)
Futurewei Technologies Inc.
2330 Central Expy
Santa Clara 95050
USA

Email: tte+ietf@cs.fau.de

Gregory Cauchie
Bouygues Telecom

Email: GCAUCHIE@bouyguestelecom.fr

Wolfgang Braun
University of Tuebingen

Email: wolfgang.braun@uni-tuebingen.de

Michael Menth
University of Tuebingen

Email: menth@uni-tuebingen.de

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: September 6, 2018

T. Eckert
Huawei
G. Cauchie
Bouygues Telecom
W. Braun
M. Menth
University of Tuebingen
March 5, 2018

Protection Methods for BIER-TE
draft-eckert-bier-te-frr-03

Abstract

This document proposes protection methods for the BIER-TE architecture [I-D.ietf-bier-te-arch]. These include 1+1 (live-live) path/tree [RFC7431] redundancy, 1:1 path/tree protection, as well as fast reroute (FRR) methods. The latter may protect against link and/or node failures and leverage infrastructure tunnels, BIER-in-BIER encapsulation, or header modification for implementation.

In particular, this memo describes FRR for BIER-TE in detail. FRR for BIER-TE requires support from the BIER-TE controller and the BFRs that are attached to a link/adjacency for which FRR protection is desired. FRR relies on the BIER header described in [RFC8279] which is also used by BIER-TE. It does not require extensions or modifications to existing BIER-TE tables. However, the presented FRR procedures need some additional forwarding plane logic on the BFR. An additional table is needed that carries information about pre-computed backup paths. When a failure is detected, the information from this table is used to modify the bitstring in the BIER header before forwarding a packet over a backup path. To prevent undesired packet duplication, packets should be tunneled on the backup paths.

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 6, 2018.

Copyright Notice

Table of Contents

1. Introduction 3
2. Overview of FRR Mechanisms for BIER-TE 3
2.1. Path/Tree Diversity 3
2.1.1. 1+1 Path/Tree Redundancy 3
2.1.2. 1:1 Path/Tree Protection 5
2.2. 1:1 Link/Node Protection 5
2.2.1. Link Protection using Existing Mechanisms 6
2.2.2. Node Protection using Existing Mechanisms 6
2.2.3. Native 1:1 Protection using BIER-in-BIER Encapsulation 7
2.2.4. Native BIER-TE Node and Link Protection 7
3. FRR Extension for Native 1:1 Protection with BIER-TE 7
3.1. FRR Key Concepts 8
3.2. The BIER-TE Adjacency FRR Table (BTAFT) 9
3.3. FRR in BIER-TE Forwarding 10
3.4. FRR in the BIER-TE Controller 10
3.5. BIER-TE FRR Benefits 11
3.6. Adjustment to the BIER-TE Forwarding Pseudocode 11
3.7. Recommendations for Tunneling 14
4. IANA Considerations 14
5. Acknowledgements 14
6. Change log [RFC Editor: Please remove] 14
7. References 15
7.1. Normative References 15
7.2. Informational References 15
Authors' Addresses 15

1. Introduction

The BIER-TE architecture is defined in [I-D.ietf-bier-te-arch] and does not provide any protection mechanisms. However, there are several approaches to protect multicast traffic against network failures. This draft gives an overview of 1+1 (live-live) path/tree redundancy, 1:1 path/tree protection, as well as fast reroute (FRR) methods including link and node protection. They may leverage either existing mechanisms with some extensions for BIER-TE, BIER-TE point-to-multipoint tunnels, or renounce on tunneling at all with the downside of reduced coverage.

This document describes additions to the BIER-TE architecture that facilitate FRR for link and node protection using BIER-TE encapsulation. Similar extensions are needed for node-protection FRR with existing mechanisms and must be supported by and must be supported by the BIER-TE controller and BFRs attached to a link/ adjacency for which FRR support is required. The FRR operation modifies the BIER header to facilitate local bypass of failed elements. It is possible to add and remove multicast links to the header, use unicast tunnels, e.g., MPLS tunnels, to implement the bypass, or to leverage BIER-in-BIER encapsulation.

The BIER-TE FRR method only requires a small set of additional forwarding entries that are stored in a separate table called the BTAFT. The entries depend only on the topology and not on the multicast flows.

2. Overview of FRR Mechanisms for BIER-TE

In the following we give an overview of various protection methods that may be applied to protect BIER-TE-based multicast trees.

BIER-TE can be combined with a range of mechanisms to provide resilience in the face of failures such as link or nodes. In this section, we give an overview of the key options.

2.1. Path/Tree Diversity

2.1.1. 1+1 Path/Tree Redundancy

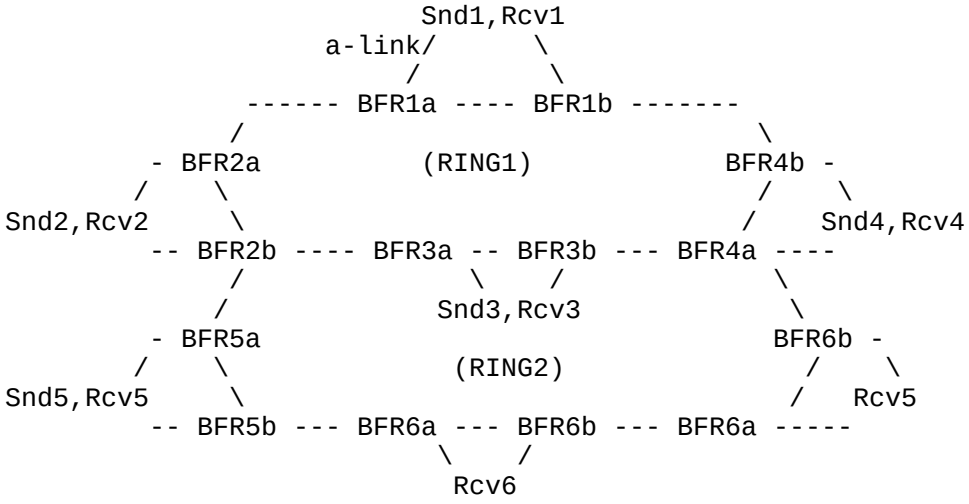
In 1+1 redundancy, often also called live-live, traffic is sent twice across the network, path-engineered in a way that no single point of failure (link or node) is in the path for both copies towards every single receiver. For point-to-multipoint transmission, this basically requires disjoint ent-to-end paths. For point-to-multipoint transmission, distribution structures are needed that fulfill the earlier mentioned criterion. They usually resemble

"orthogonal" tree structures. See [BIERFRRANALYSIS] for illustration. For multicast transmission, 1+1 redundancy can be very attractive because the cost of dual transmission can often be negligible vs. the overall bandwidth saving of doing replication in the network.

Path-engineering for 1+1 redundancy can become quite advanced depending on the topology - but it does not require any new functionalities from BIER-TE. It just requires appropriate engineering and potentially additional application or BIER edge functions such as traffic duplication at the sender and traffic deduplication at the receiver.

Consider the following topology example: each sender or receiver has connections to two BFRs to overcome the failure of either BFR. The application on the source creates two copies for every packet. It sends one "a-packet" copy onto its a-link and the other "b-packet" copy onto its b-link. Snd,Rcvr could be BFIR,BFER or they could send native multicast and the BFR they connect to are BFIR, BFER.

BIER-TE bitstrings need to be set up for a-packets and b-packets to pass through the topology counter-circular to each other so that any individual link or node failure never interrupt both a-packets and b-packets



The application side in Snd,Rcv needs to be able to eliminate the duplication resulting from receiving both a-packet and b-packet copies (unless there is a failure). This is easily done if there is any encapsulation (such as transport layer) where sequence numbers are used. Otherwise such a layer has to be added.

If sender and/or receivers can not support the duplication on the sending side and/or deduplication on the receiving side, it is easy to derive variations of these designs in which network ingress/egress devices take over these rules. The functionality that is least common in network devices is per-packet deduplication based on sequence numbers in the packets. Only few network transport service encapsulations do currently provide sequence numbers, e.g., L2TPv3.

Because the described ingress/egress functionalities are not specific to BIER-TE but would equally apply to the solution if built with native IP multicast or RSVP-TE/P2MP, this document does not propose any further functionality required for this type of solutions.

Compared to RSVP-TE/P2MP, one specific benefit of BIER-TE is that trees can be optimized without re-signaling solely by the BIER-TE sender adding/deleting bits representing leaf trees to receivers that join/leave the content.

2.1.2. 1:1 Path/Tree Protection

If duplicate transmission is not desirable, variations of the above scheme can be devised where only one copy is sent to each receiver. Such solutions require receiver feedback to the sender and are therefore most feasible for a limited receiver set deployment, e.g., regional multi-system operator (MSO) distribution networks to hybrid fiber-coaxial (HFC) headends. For example, by default only the a-tree could be transmitted, and upon reception failure at any subset of receivers, the sender would transmit to the subset of the b-tree covering those receivers. Upon recovery of the a-tree, signaling from the receiver could equally stop transmission of the b-tree toward this receiver.

For a more common path utilization across the network in the no-failure scenario, senders could instead start with 50% receiver populated a-tree and b-tree as well.

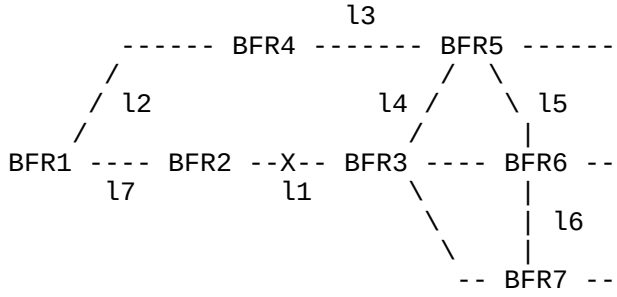
These options heavily depend on the ability to change the set of receivers in a tree without signaling. The enabling/disabling of sending to a particular branch of the network can be done within a single round-trip starting with the signaling of the reception failure by a receiver.

2.2. 1:1 Link/Node Protection

In this section we discuss the link and node protection for BIER-TE using existing mechanisms and the native BIER-Te FRR approach.

2.2.1. Link Protection using Existing Mechanisms

BIER-TE can be used in conjunction with reactive 1:1 link protection as it is also possible in RSVP-TE/P2MP. When a node sees a failure on a link, it assumes that the link has failed and uses a pre-established backup path to get packets to the node on the other side of the link. In example below, BFR2 would use a pre-established path via l7,l2,l3,l4 to send packets to BFR3 when it sees a failure of link1.



In BIER-TE, this link protection can be done either by using some pre-existing traffic engineered tunneling mechanisms such as RSVP-TE/P2P or Segment Routing paths to get packets to BFR3 in the link failure case. These options do not require enhancements to BIER-TE.

2.2.2. Node Protection using Existing Mechanisms

BFR2 can not distinguish whether link1 or BFR3 fails. It simply needs to assume one or the other and perform link or node protection. If it assumes node failure and wants to perform node protection, the solution becomes more complex. Effectively, BFR2 needs to get the packets over to BFR5,BFR6 and BFR7 or the subset of those next-next-hops that is required. Using pre-established p2p backup tunnels (RSVP-TE/P2P or SR) allows to send just to the required subset of next-next-hops at the expense of sending the traffic up to three times across the path consisting of the links l7,l2,l3. The required subset of next-next-hops consists of the downstream next-next-hops. Downstream means that the next-next-hops are the direct multicast children of the next-hop. This is similar to the FRR concept discussed in Section 3.1. Using a backup RSVP-TE/P2MP tree eliminates this higher load but would deliver to all next-next-hops or it would be necessary to set up backup RSVP-TE/P2MP trees for all possible subsets of next-next-hops (which may not scale well).

2.2.3. Native 1:1 Protection using BIER-in-BIER Encapsulation

A simpler solution for node protection leverages BIER-in-BIER encapsulation. In this approach, BFR2 would encapsulate the BIER-TE packet into another BIER-TE packet whose bitset was pre-built to carry the packet via l7,l2,l3,l5,l6 to BFR5,BFR6,BFR7 optionally reducing the bitset based on the bitset of the encapsulated BIER-TE packet. This document describes this option in Section 3.

If there are no available infrastructure tunnels deployed, then BIER-in-BIER encapsulation could equally be used for the simple case of link protection.

2.2.4. Native BIER-TE Node and Link Protection

An even simpler solution is to not encapsulate the BIER-TE packet into another BIER-TE packet but instead to rewrite the bitstring of the BIER-TE packet to make the packets get delivered via l2,l3 to BFR5, via l5 to BFR6 if that is part of the original bitstring and via l6 to BFR7, if that is part of the original bitstring.

This document specifies the necessary rules for the BIER-TE forwarding machinery for such bitstring bitstring rewrites to enable the most lightweight and efficient form of node protection with BIER-TE.

This solution will not work in all topologies though because the set of bits necessary to pass the traffic across a backup path/tree may cause undesired traffic forwarding (duplicates/loops).

3. FRR Extension for Native 1:1 Protection with BIER-TE

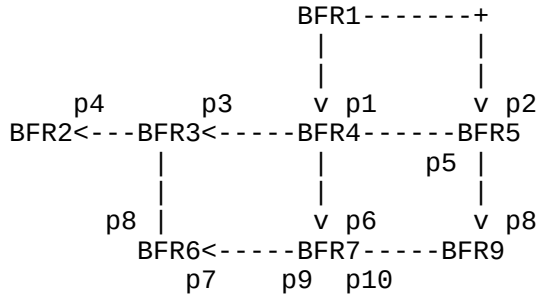
In this section, we explain the FRR extension to support native 1:1 protection with BIER-TE. These extensions do not require changes to the BIER header or to forwarding mechanism. The protection procedure runs directly before the BIER-TE forwarding procedure.

Note that the FRR extensions require the use of a new table which is described in Section 3.2. The table is only filled with entries that depend on the network topology and do not depend on the multicast flows.

An implementation of BIER-TE FRR based on BIER-in-BIER encapsulation in the networking programming language P4 is given in [BIER-FRR-P4]. The source code is thoroughly documented and contains examples how the BIER-TE BIFT and BIER-TE BTAFT tables can be filled.

3.1. FRR Key Concepts

In this section we use the following example to explain the key concepts of BIER-TE FRR. The example shows a multicast tree from BFR1 to BFR2, BFR6, BFR9. The path to BFR2 is represented by the bits p1, p3 and p4. The bits p1, p6, p7 and the bits p2, p8 represent the path towards BFR6 and BFR 9, respectively. Local_decap bits for all BFR2,BFR6, and BFR9 are also used.



First, we consider that the link from P towards F fails. The failure can be protected by the backup paths over BFR3->BFR6->BFR7: p3, p8, p9 (BP1) and BFR5->BFR9->BFR7: p5, p8, p10 (BP2). The use of backup path BP1 does not cause duplicates. Backup path BP2 would cause duplicates because the local_decap bit for D2 is still set in bitstring at P. Two options exist to avoid duplicates.

1. We reset the local_decap bit for D2: This solution prevents the duplicate packet. However, this method can lead to lost packets in other examples.

2. We use a tunnel from P to F over D2 to prevent BIER packet processing at the nodes at the backup path.

Tunnels may be implemented in these two different ways:

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

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

Now, we consider that BFR7 fails. The backup path must send the packets to all downstream next next-hops (DS-NNHs), i.e. the next-hops of the sub-tree rooted of BFR7. BFR4 can identify the DS-NNHs by checking the bits of interest of the failed node BFR7. BFR6 is such a node because bit p7 is set. BFR9 is not downstream because there is no bit of interest from BFR7 to BFR9. Sending packets to BFR9 would causes duplicates because BFR9 is served using the branch BFR1->BFR5->BFR9.

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

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

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

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

...

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

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

The failed BitPosition is the BP in the BIFT in which the FRR Adjacency is used. The downstream BitPosition is required to protect against node failures to identify the downstream adjacency as described in Section 3.1. The backup path/tree is constructed of the individual ResetBitmasks and AddBitmasks of the downstream nodes. To protect against link failures, the DS-BP field is set equally to the F-BP field.

3.3. FRR in BIER-TE Forwarding

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

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

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

3.4. FRR in the BIER-TE Controller

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

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

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

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

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

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

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

3.5. BIER-TE FRR Benefits

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

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

- For the case of a node failure, it allows to build a path engineered backup tree as opposed to a set of partly overlapping p2p backup tunnels.

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

3.6. Adjustment to the BIER-TE Forwarding Pseudocode

We augment the forwarding procedure presented in the BIER-TE draft to support FRR.

The following procedure computes the ResetBitmasks and AddBitmasks when an adjacency up/down notification is triggered. The masks can later be directly applied to the header to facilitate the backup.

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

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

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

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

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

The ForwardBierTePacket procedure must be modified by applying the FRR operations when necessary.

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

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

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

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

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

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

            PacketCopy = Copy(Packet)

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

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

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

3.7. Recommendations for Tunneling

Recommendations for usage of tunnels for BIER-TE FRR based on the study in [BIERFRRANALYSIS]:

1. Tunneling SHOULD be used. The study shows that header modification does not improve coverage in many topologies and may cause more harm than protection when node failures occur.

2. Using unicast tunnels may cause unnecessary extra load on overlapping links when protecting against node failures. Moreover, they require additional bits in the BIER header.

3. BIER-in-BIER encapsulation is the recommended mechanism. It causes increased header overhead through the encapsulating BIER header. This may become an issue when the BIER header supports long bitstrings

4. IANA Considerations

This document requests no action by IANA.

5. Acknowledgements

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

6. Change log [RFC Editor: Please remove]

03: Updated references, no other conent changes from -02. Refresh to continue using this document for reference of new work. Unchanged. Currently unclear if the novel method proposed in this document is worth pursuing. If not, then we would change the document to solely document the applicability of pre-existing methods (and remove new options).

02: Overview of FRR mechanisms feasible for BIER-TE (Eckert)

02: Removed Section "BIER-TE and existing FRR, because it is now part of "Overview" chapter (Braun)

02: Added recommendation on native BIER-TE FRR with regard to tunneling options based on a study (Braun)

02: Added P4 implementation of BIER-TE FRR using BIER-in-BIER as reference.(Braun)

01: Update of author addresses

00: Initial version based on draft-eckert-bier-arch-03.

7. References

7.1. Normative References

[I-D.ietf-bier-te-arch]
Eckert, T., Cauchie, G., Braun, W., and M. Menth, "Traffic Engineering for Bit Index Explicit Replication (BIER-TE)", draft-ietf-bier-te-arch-00 (work in progress), January 2018.

[RFC7431] Karan, A., Filsfils, C., Wijnands, IJ., Ed., and B. Decraene, "Multicast-Only Fast Reroute", RFC 7431, DOI 10.17487/RFC7431, August 2015, <https://www.rfc-editor.org/info/rfc7431>.

[RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast Using Bit Index Explicit Replication (BIER)", RFC 8279, DOI 10.17487/RFC8279, November 2017, <https://www.rfc-editor.org/info/rfc8279>.

7.2. Informational References

[BIER-FRR-P4]
Braun, W., Hartmann, J., and M. Menth, "Demo: Scalable and Reliable Software-Defined Multicast with BIER and P4", IFIP/IEEE International Symposium on Integrated Network Management (IM), May 2017, <https://atlas.informatik.uni-tuebingen.de/~menth/papers/Menth17b.pdf>.

[BIERFRRANALYSIS]
Braun, W., Eckert, T., and M. Menth, "Performance Comparison of Resilience Mechanisms for Stateless Multicast Using BIER", IFIP/IEEE International Symposium on Integrated Network Management (IM), May 2017, <https://atlas.informatik.uni-tuebingen.de/~menth/papers/Menth17a.pdf>.

Authors' Addresses

Toerless Eckert
Huawei USA - Futurewei Technologies Inc.
2330 Central Expy
Santa Clara 95050
USA

Email: tte+ietf@cs.fau.de

Gregory Cauchie
Bouygues Telecom

Email: GCAUCHIE@bouyguestelecom.fr

Wolfgang Braun
University of Tuebingen

Email: wolfgang.braun@uni-tuebingen.de

Michael Menth
University of Tuebingen

Email: menth@uni-tuebingen.de

BIER Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2017

IJ. Wijnands
P. Pfister
Cisco Systems
J. Zhang
Juniper Networks

July 8, 2016

Generic Multicast Router Election on LAN's
draft-wijnands-bier-mld-lan-election-01.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 January 9, 2017.

Copyright Notice

Table of Contents

1. Introduction 2
2. Terminology and Definitions 3
3. Specification of Requirements 4
4. Problem Statement 4
4.1. Receiver side 4
4.2. Sender side 5
5. Proposal 6
6. DF Election Mechanism Requirements 7
7. The DF Election mechanism 8
7.1. Highest Random Weight 8
7.2. The DF Hello Message 8
7.3. The Designated Announcer 9
7.3.1. DAL Hello Option 9
7.3.2. A new Candidate DF 9
7.3.3. A candidate DF goes down 10
7.4. DA Inconsistency 11
8. The Hello Message Packet Format 11
9. Security Considerations 11
10. IANA Considerations 12
11. Acknowledgments 12
12. Normative References 12
Authors' Addresses 13

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.

DF:
A Designated Forwarder is responsible for accepting a multicast packet from a LAN.

DR:
A Designated Router is responsible for forwarding a multicast packet onto a LAN.

DA:
A Designated Announcer is a router that is responsible for announcing a list of candidate Designated Forwarders.

DAL:
A Designated Announcer List is generated by the DA and holds the candidate Designated Forwarders.

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.

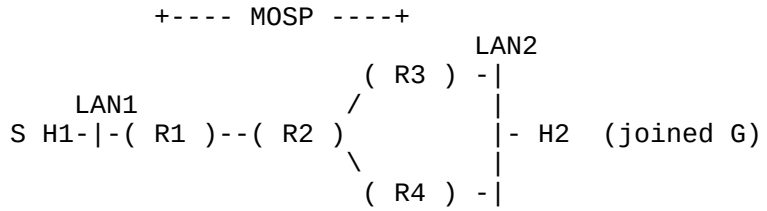


Figure 1

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:

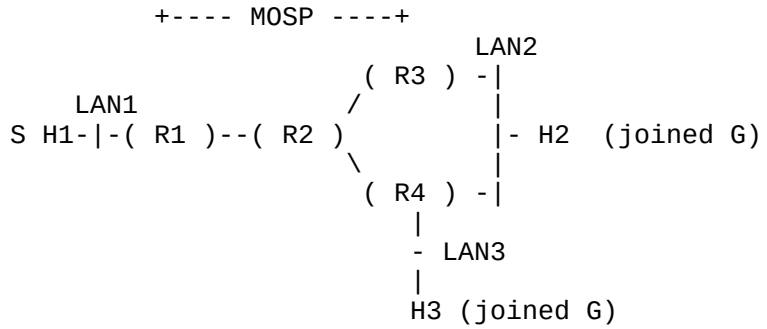


Figure 2

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.

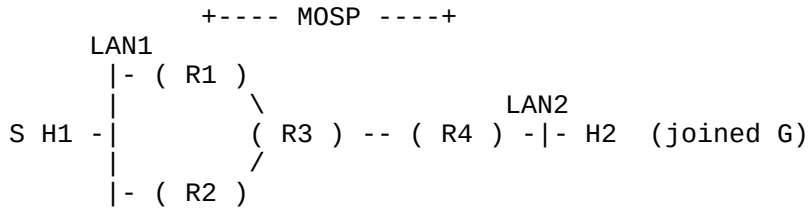


Figure 3

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 sam LAN. 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 network 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 by 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 for Group membership discovery, 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 for 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 election mechanism for the DR and the DF is very similar. In fact, when a DF is elected, it MUST always be used as the DR as well to avoid multicast packet looping. The procedures in this document always elect a DF on the LAN, and for that reason will always be the DR. In the sections that follow, we don’t refer to the DR anymore. Everywhere where we reference DF, we implicitly mean it applies to both the DR and DF.

6. DF Election Mechanism Requirements

When electing a DF on the LAN, it is important to have a single DF for a given Multicast flow at all times. If during the election process (or changes to it), there is no DF, it will cause traffic loss to the end user. If there are two (or more) DFs at the same time, it may cause traffic duplication or even loops. Since the election is done among different routers, it is not so trivial to guarantee that there will never be inconsistency in the DF election. There is also a tradeoff between the complexity introduced and the incremental benefit it brings. The procedures in this document are designed to detect inconsistency and recover from it as fast as possible. During inconsistency, we prefer traffic loss over possible duplication or looping of multicast packets.

When there are multiple candidate DF routers on the LAN, it is beneficial to load-balance the traffic over the different candidate DFs. This helps to distribute the bandwidth usage among the routers, reduce the impact of a router failure and shorten the failover time when changing the DF for effected flows. For that reason the DF procedures MUST support DF election per multicast group address.

7. The DF Election mechanism

7.1. Highest Random Weight

The method proposed to select a DF is based on the Highest Random Weight (HRM) as described in [RFC2291]. The paragraph below is mostly taken (and modified) from [RFC2291].

The router computes the weight for EACH candidate DF by performing a hash over the Group address that identifies the flow, as well as over the address of the candidate DF. The router then chooses the candidate DF with the highest resulting weight value. This has the advantage of minimizing the number of flows affected by a candidate DF addition or deletion (only 1/N of them), but is approximately N times as expensive as a modulo-N hash.

In order to get a good distribution of the Group addresses over the candidate DFs, it is important we choose a good Pseudorandom function to calculate the Weight. The Weight is calculated using the Group (G) IP address and the Candidate DF (CDF) IP address.

Weight(G, CDF) =
(1103515245((1103515245.G+12345)XOR CDF)+12345)(mod 2^31)

If multiple Candidate DFs end up with the same highest weight, the DF with the lowest IP address MUST be selected.

If every candidate DF on the LAN uses the same HRW algorithm to select the DF for a particular Group out of the same list of candidate DFs, they all will reach the same conclusion and there will be no inconsistency. It is very important every router on the LAN has the same list of candidate DFs. The mechanism proposed in this draft to generate a consistent list is based on the new Hello message.

7.2. The DF Hello Message

In order to discover the candidate DFs we need a mechanism to learn them. We introduce a new (IGMP/MLD) message type called the DF Hello. Routers on a LAN that are candidate DFs periodically send DF Hellos. The message format is specified later in a later revision document. Based on the DF Hellos it is possible to generate a list of candidate DFs. However, it is challenging to keep the candidate DF list synchronized between the routers when DFs are added or removed from the list as each router will do that based on its own scheduling. Especially when candidate DFs timeout, it is very likely this happens at different times and opens up the opportunity for inconsistency. Also, when a new candidate DF is added to the network

and one of the routers did not get the initial DF Hello message, its candidate DF list will be out of sync until the next DF Hello is received, leading to a inconsistent candidate DF list for a relatively long period. In order to help synchronize the candidate DF List we elect a Designated Announcer (DA).

7.3. The Designated Announcer

The router that will act as the Designated Announcer is determined by the Priority value as included in the Hello message, using the IP address as tiebreaker. The router with the highest priority is preferred, if there are multiple routers with the same priority, the router with the highest IP address is preferred. The DA determines which routers from the Hello List (HL) are included in the Designated Announcer List (DAL). By default all the routers in the HL are considered to be included in the DAL. It is however possible to filter certain candidates and not include these in the list based on some sort of preference.

7.3.1. DAL Hello Option

The DAL is sent out by the DA as an Option included in its Hello message. In order to reliably transmit the Hello Message with the DAL option, a DAL sequence number is included in the packet along with an acknowledgement flag for each router in the DAL. Every router in the DAL MUST respond by triggering a Hello message including this sequence number. If the DA has not received a response within a given timeout from certain routers in the DAL it will re-transmit the Hello message with the Acknowledgement flag not set for the routers that have not responded. The routers on the LAN that see their IP address in the DAL without the acknowledgement flag set will re-transmit their Hello. This process continues until the DA has received a response from all the routers in the DAL. Using this mechanism we minimize the time an inconsistency can occur when a router has missed a Hello message that includes that DAL.

7.3.2. A new Candidate DF

When a new candidate DF becomes active on the LAN, it first has to learn if there are other candidate DFs on the LAN. Learning about other candidate DFs is accelerated by setting the Learn Flag in the Hello message. Routers on the LAN that receive a Hello with the Learn Flag set will trigger a Hello message in response. After the learning delay the new DF assumes all candidate DFs on the LAN have responded and the Hello List is complete. There are three different scenarios the new DF has to consider.

7.3.2.1. The Hello List is empty

When the HL is empty, the new DF will become the DA with only its own address in the DAL. The DF will start to act as DF for all the groups.

7.3.2.2. The New DF is not the DA

When there are other candidate DFs on the LAN, the Hello List is populated. If the new DF is not the DA, it will have to wait for the DA to include its address in the DAL. As soon as it sees its own address in the ADA with the acknowledgement flag not set, it will trigger a Hello message with the DAL sequence number and start to act as DF. Note, it is likely that new DFs IP address is already included in the first Hello message it receives from the DA.

7.3.2.3. The New DF is the DA

After the Learning delay the new router may find it self having the highest Priority and will be the new ADA. Note, we prefer the DA to be deterministic so the new DF will take over the role of the DA. The DF which is currently the DA will have seen the Hello message from the new DF and will realize this is the new DA. The current DA MUST respond by sending a Hello message without the DAL in it. All the routers on the LAN will now know that the current DA is going away. The candidate DFs MAY continue to use the old DAL until the new DAL list is received from the new DA. The new DF will create the DAL list based on its Hello List and send out a Hello message, following the procedures as described above. If during a transition of the DA a router detects inconsistency between the received DAL and the perceived DA, the router stops using the current DAL and waits until the inconsistency is resolved. This inconsistency may have occurred due to missing a DF Hello message (also see section DA inconsistency).

7.3.3. A candidate DF goes down

When a DF goes down there are 2 different scenarios to consider.

7.3.3.1. The DF was the DA

When a DF goes down, due to a failure or an operator removing it from the LAN, the routers on the LAN will eventually detect this because the Holdtime for that DF will expire. This does not have an immediate effect on the DF procedures because the DF is chosen from the DAL, originated by the DA. A candidate DF MUST NOT take any action based on a candidate DF going down, but MUST wait for the DA to sent out a new DAL list. This will ensure that all candidate DFs

on the LAN will start to use the new DAL at the same time and avoid any discrepancies due to routers expiring the timer associated with the DF that went down.

7.3.3.2. The DF was the DA

If the DF that goes down is the DA, a new DA has to be elected. Note, every candidate DF on the LAN is a potential candidate to become the new DA. The new DA is chosen based on the Hello List using the Designated Announcer election procedures. It is possible a candidate DF receives the DAL from the new DA before it detected the current DA is down. This may be due to a race condition where timers on the candidate DF expire at different times. We use the procedures as described in section (DA inconsistency).

7.4. DA Inconsistency

A candidate DF that receives a DAL from a router that it does not consider to be the active DA MUST immediately stop acting as a DF. The candidate DF MUST wait for the DA inconsistency to be resolved before it is allowed to resume its role as candidate DF. This will cause traffic to be blocked for the multicast groups this DF is responsible for, but it will not cause traffic duplication and/or loops due to other DFs using a different DAL list. The inconsistency can be resolved due to the following events.

- The active DA expires.
- A Hello is received from the active DA without a DAL.

When the candidate DF detects that there is only one candidate DF that has announced the DAL and it is considered to be the DA, the inconsistency is resolved and the DF can resume its role as DF for the Groups it is responsible for.

8. The Hello Message Packet Format

The format of the Hello Message is included on the next revision of this document.

9. Security Considerations

TBD.

10. IANA Considerations

TBD.

11. Acknowledgments

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

12. 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>.

[RFC2291] Slein, J., Vitali, F., Whitehead, E., and D. Durand, "Requirements for a Distributed Authoring and Versioning Protocol for the World Wide Web", RFC 2291, DOI 10.17487/RFC2291, February 1998, <http://www.rfc-editor.org/info/rfc2291>.

[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

Jeffrey Zhang
Juniper Networks
10 Technology Park Dr.
Westford MA 01886
US

Email: zzhang@juniper.net