I:

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

T. Eckert
Cisco Systems, Inc.
G. Cauchie
Bouygues Telecom
W. Braun
M. Menth
University of Tuebingen
July 8, 2016

Fast ReRoute (FRR) Extensions for BIER-TE
draft-eckert-bier-te-frr-00

Abstract

This document proposes an Fast ReRoute (FRR) extension to the BIER-TE Architecture [I-D.eckert-bier-te-arch]. The FRR procedure has to be supported by the BIER-TE Controller host and the BFRs that are attached to a link/adjacency for which FRR support is required. Thus, the FRR concept can be incrementally deployed in the data plane to only those BFR adjacent to adjacencies for which FRR protection is desired.

The FRR procedure does not require changes to the packet format described in [I-D.ietf-bier-architecture] that is also used for BIER-TE. Existing BIER-TE tables do not have to be altered. FRR procedures do require additional forwarding plane logic on the BFR that need to support FRR.

An additional table is needed that carries information about pre-computed backup paths. This table is used to modify upon detection of failure the bitstring in the BIER header. To prevent packet duplicates, tunneling mechanisms such as remote adjacencies or BIER-in-BIER encapsulation can be leveraged.

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.

Table of Contents

1. Introduction

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

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

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

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

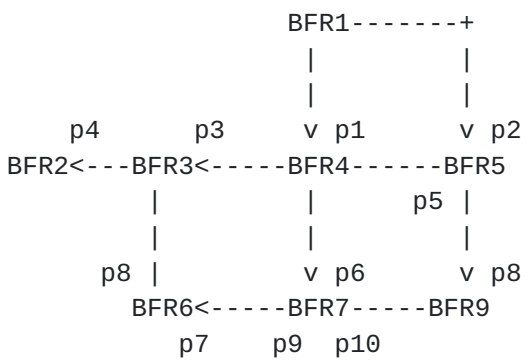
FRR is an optional procedure because it does require additional control plane, and forwarding plane and not all BIER-TE networks may want to use it. Alternatives to FRR include the following:

"Live-Live" - transmitting the same traffic twice across two BIER-TE engineered diverse paths. Live-Live is popular in deployments where actual receiver equipment can already deal with dual reception (eg: SMPTE ST 2022-7 seamless protection switching in video system). Likewise, MoFRR (Multicast only Fast Reroute, RFC 7431 [RFC7431]) could be used on BFER to merge traffic from two TE engineered diverse paths for receivers that can not deal with dual-reception.

BFIR FRR: Because BIER-TE is stateless, it is feasible to consider simply changing the bitstring on a BFIR upon detection of a failure. Such an approach would require fast propagation of detected failures, pre-calculation or fast-inline-calculation of the modified bitstrings and then quickly pushing these into the BFIR. Due to the absence of statelessness in solutions preceeding BIER-TE there are no good data points what performance could be achieved from such an approach yet in various network/tree setups.

2. FRR Key Concepts

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



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

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

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

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

Protection against link failures only requires knowledge of the failed adjacency. Protection against node failures requires additional knowledge of the downstream nodes of the tree. The

computation of appropriate backup paths, AddBitmasks, ResetBitmasks, and BitPositions is outside of the scope of this document.

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

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

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

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

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

The failed BitPosition is the BP in the BIFT in which the FRR Adjacency is used. The downstream BitPosition is required to protect against node failures to identify the downstream adjacency as described in Section 2. 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.

4. FRR in BIER-TE forwarding

The BIER-TE forwarding plane receives fast Up/Down notifications of BIER adjacencies which are used to with the FRR Adjacencies Index for

different SIs. From the failed BitPosition in the BTAFT entry, it remembers which BPs are currently affected (have a down adjacency).

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

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

5. FRR in the BIER-TE Controller Host

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

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

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

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

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

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

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

6. BIER-TE FRR Benefits

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

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

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

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

7. 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 Reset- and AddBitmaks when a 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)
}
```

8. BIER-TE and existing FRR

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

If BIER-TE FRR is not feasible or necessary, it is also possible for BIER-TE to leverage any existing form of "link" protection. For example: instead of directly setting up a forward_connected adjacency to a next-hop neighbor, this can be a "protected" adjacency that is maintained by RSVP-TE (or another FRR mechanism) and passes via a backup path if the link fails.

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

9. IANA Considerations

This document requests no action by IANA.

10. Acknowledgements

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

11. Change log [RFC Editor: Please remove]

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

12. References

[I-D.eckert-bier-te-arch]
Eckert, T., Cauchie, G., Braun, W., and M. Menth, "Traffic Engineering for Bit Index Explicit Replication BIER-TE", draft-eckert-bier-te-arch-03 (work in progress), March 2016.

[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-03 (work in progress), January 2016.

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

Authors' Addresses

Toerless Eckert
Cisco Systems, Inc.

Email: eckert@cisco.com

Gregory Cauchie
Bouygues Telecom

Email: GCAUCHIE@bouyguestelecom.fr

Wolfgang Braun
University of Tuebingen

Email: wolfgang.braun@uni-tuebingen.de

Michael Menth
University of Tuebingen

Email: menth@uni-tuebingen.de