BIER-TE-ARCH
IETF104 Prague

draft-ietf-bier-te-arch-01

Toerless Eckert (tte@cs.fau.de)
Gregory Cauchie (GCAUCHIE@bouyguestelecom.fr)
Wolfgang Braun (wolfgang.braun@uni-tuebingen.de)
Michael Menth (menth@uni-tuebingen.de)

Since IETF103

- No further version since IETF103
  - Version -01 from IEF103 introduced proposed requirements against forwarding plane
    - MUST for most important forwarding options, SHOULD, MAY for more complex options
    - Says RFC8926 can be used unchanged for BIER-TE (no new encap needed)
    - Reworked details of comparison with BIER - offering two comparison options

- Authors think document is stable
  - Ask for working group last call after IETF104
  - Reminder: target is EXPERIMENTAL
  - Just not enough deployment experience
    - Deployable today only via CLI hop-by-hop config
    - Next highest priority: work on YANG model (interesting discuss during IETF103)

THE END

Slides from IETF103 as backup in case further discussion wanted

3.6 Requirements

- New section with MUST/SHOULD against forwarding plane to comply
  - Similarily to BIER architecture where ECMP is optional, mandate with MUST the most core BIER-TE forwarding functions, SHOULD/MAY the others
- MUST
  - Configure subdomain for BIER-TE forwarding
  - Bits indicating no or one adjacency: forward connected, forward routed or local_decap
    - Forward_routed could be degraded, but for scalability we think its very important
    - Aka: adjacency is encap into another label to send to remote BIER-TE router
- SHOULD
  - DNR (Do Not Reset) flag on adjacencies. To support L3 rings with just one bit per direction.
- MAY
  - ECMP adjacencies.
    - Only MAY (less important than in BIER), because feeling is that with TE you want to manage amount of traffic on paths more explicit, and doing that with ECMP is more difficult than explicit, non-ECMP paths.
  - More than one adjacency on a bit. Flood from hub to multiple spoke adjacencies. Good for broadcast TV style traffic, also sounds like on the bottom of the priority list.

BIER pseudocode adopted to BIER-TE:

```
void ForwardBitMaskPacket_withTE (Packet)
{
   SI=GetPacketSI(Packet);
   Offset=SI*BitStringLength;
   for (Index = GetFirstBitPosition(Packet->BitString); Index ;
        Index = GetNextBitPosition(Packet->BitString, Index)) {
      F-BM = BIFT[Index+Offset]->F-BM;
      if (!F-BM) continue;
      BFR-NBR = BIFT[Index+Offset]->BFR-NBR;
      PacketCopy = Copy(Packet);
      PacketCopy->BitString &= F-BM;              [2]
      PacketSend(PacketCopy, BFR-NBR);
      // The following must not be done for BIER-TE:
      // Packet->BitString &= ~F-BM;              [1]
   }
}
```

- Same pseudocode than BIER
  - Just commented out [1]
- Doing [1] and [2] together in BIER-TE does not work
- You want to reset the bit B you are forwarding on in [2], so
  - F-BM = ~ (2 << B)
- But then [1] would reset all the bits.
- Simple solution: Just do not do [1] when BIFT uses BIER-TE mode

```
void ForwardBitMaskPacket_withTE (Packet)    {
  SI=GetPacketSI(Packet);
  Offset=SI*BitStringLength;
  AdjacentBitstring = Packet->BitString &= ~AdjacentBits[SI];
  Packet->BitString &= AdjacentBits[SI];
  for (Index = GetFirstBitPosition(AdjacentBits); Index ;
       Index = GetNextBitPosition(AdjacentBits, Index)) {
     foreach adjacency BIFT[Index+Offset] {
       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);  [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);     }
} } }
```

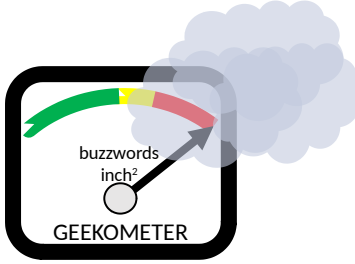
- Simplified original pseudocode from -00 version
- Removes use of FBM
  - Implementations do not need to provide FBM for BIER-TE, that's just a choice if commin HW for BIER/BIER-TE makes that beneficial
  - Only need to reset bit of adjacency itself (unless DNR set) [1]
- Also shows handling of the different adjacency types
  - No equivalent level of detail in BIER arch RFC pseudocode, but hopefully useful.

What else ?

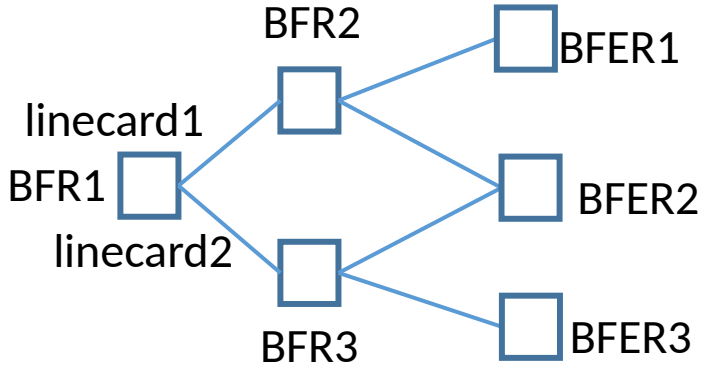
- Added node how BIER/BIER-TE might best be understood by someone coming from SR
  - BIER are 1 bit equivalent of destination/Node SIDs
  - BIER-TE could be seen as doing for multicast what an SR SID-stack would do to do explicit routing. Every hop SID is just a bit.

- Purely opportunistic text
  - Hope it is helpfull for people coming from SR looking at BIER
  - If not, will just remove (1 paragraph).

While trying to figure out BIFT for BIER-TE



- Independent of BIER-TE, but BIER thought
- Do BIER BIFT implementations have to accept arbitrary F-BM values ?
- Not currently ? There is no API from IETF (YANG,..) that allows to program BIFT/F-BM directly ?!
- BIFT/F-BM only programmed internally from BIRT ?!
- Arbitrary F-BM (not derived from BIRT) might not be possible/desirable to support in optimized BIFT
- Could easily create “funny” FBM that stretch platforms
  - E.g.: BIRT derived BIFT can be parallelized
  - Egres linecards would only need to look/examine bits with NBRs on the interface
  - Funny F-BMs would not allow to do this. ”Last” linecard may need to still evaluate impact of all preceding bits.
- Define in any BIER rev constraints on required F-BM to ensure platform optimizations can be done without running into future surprises ?
  - When BIFT ae exposed directly to programming



BFER1,2,3 are bit 1,2,3.
On BFR1:
BIFT[1]->F-BM = 011 BIFT[1]->BFR-NBR = 2
BIFT[2]->F-BM = 110 BIFT[2]->BFR-NBR = 3
BIFT[3]->F-BM = 100 BIFT[3]->BFR-NBR = 3