Carrier Grade Minimalist Multicast (CGM2) using Bit Index Explicit Replication (BIER) with Recursive BitString Structure (RBS) Addresses

IETF 112 / INTAREA

draft-eckert-bier-cgm2-rbs-00
Toerless Eckert (tte@cs.fau.de) Futurewei USA

Based on design/work from
Sheng Jiang, Xu Bing, Yan Shen, Meng Rui, Wan Junjie and Wang Chuang {jiangsheng|bing.xu|ynshen|mengrui|wanjunjie2|wangchuang}@huawei.com
Why INTAREA

• Why not? (is IntArea only for Unicast? ;-)

• To inform INTAREA of a great example, how further work on more intelligent (variable length) addressing helps to solve currently worked on routing problems better!

  • Some call (“locator type”) addresses forwarding instructions (i like that too)
  • IMHO whatever we call them, it is still routing
What is wrong with BIER-TE?

BIER-TE is the best multicast forwarding solution for its target constraints:

- BIER per flow/tree stateless forwarding with added stateless traffic steering...
- When we can only reuse the pre-existing BIER-TE "flat-bitstring"
- and only no...minimal changes to BIER forwarding algorithm.

But this comes at undesirable limitations / complexities:

N-bit bitstring can address only fixed set of N-M BFER (e.g.: N=256)

- Because every bit of address is statically assigned to an adjacency (e.g.; BFER)
- M is number of bits needed to 'steer' packets (BFR). Also fixed, potentially sub-part of topo.
- Same static mapping issues exist in BIER, but less severe

Result (as seen from BIER-TE draft):

- Design complexities minimizing required M bits (LAN, hub&spoke, p2p, overlay topologies, ...).
- Controller and network operator.
- Larger number of copies with different bitstrings to reach arbitrary subset of BFER in large network
- OPS and CONTROLLER COMPLEXITY, less TRAFFIC EFFICIENCY for FORWARDING PLANE SIMPLICITY.
Fix the bitstring addressing issue

Eliminate duplication: IP multicast on top of bitstring forwarding ("overlay")
- End-to-end integration with controller plane

Complete TE services via controller plane and forwarding plane
- Deterministic Latency, Path Selection (Steiner, diverse, ...)

Encapsulation format:
- **Unicast encap**
- **Multicast encap**
- **Payload**
Recursive Bitstring Structure (RBS)

- **Fixed Field**
  - Total len: indicate the total length of the Recursive Unit. Filed length can be set flexibly to 8/16/32, with bit/byte/word. Padding is not included.
  - Rsv: Reserved Fields

- **Recursive Unit**
  - Bitstring: instructs a node to locally duplicate packets and forwarding – like a BIER-TE bitstring (but without complex options)
  - Addressing Field: length of ‘child’ recursive units.
  - Recursive Unit: has the same structures as the unit it sits in.

- **Padding** (Optional): Used for alignment to byte (or larger).
**Multicast route identifier**
- Define the local bitmap.
- Each bit corresponds to a multicast object, which denote the direct delivery destination of multicast packet.

**Forwarding table**
- Each table entry corresponds to a bit in the bitmap.
- Instruct the forwarding method to the corresponding multicast object (outgoing interface, next hop address, etc.)

---

*The bug in this example is left as an exercise to the reader*
Simplifications / Performance Enhancements

Naturally no need for loop prevention measures (as in BIER/BIER-TE) (no clear bits)

No need to split up topology into different SI and/or multiple SD bitstrings

No (or much smaller) need to create optimized bit semantics.
   Just use all adjacencies of topology and/or shortest tunnel adjacencies over unicast-routers.

Especially for ‘sparse’ (small number of BFER) distributions
   Can encode any subset and steering in single RBS address == single packet packet
   If majority of packet have sparse receiver sets. Duplicate packet reduction maximized.

When total number of BFR too large for desired max RBS address structure (eg: > 512 bit ?)
   Controller can arbitrary duplicate/split distribution tree to create multiple packets for 'sub-trees'.
Forwarding Pseudocode

```c
void ForwardRBSPacket (Packet)
{
    RBS = GetPacketMulticastAddr(Packet);
    Total_len = RBS;
    Rsv = Total_len + length(Total_len);
    BitStringA = Rsv + length(Rsv);
    AddressingField = BitStringA + BIFT.entries;

    // [1] calculate number of recursive bits set in BitString
    CopyBitString(BitStringA, *RecursiveBits, BIFT.entries);
    And(*RecursiveBits, *BIFTRecursiveBits, BIFT.entries);
    N = CountBits(*RecursiveBits, BIFT.entries);

    // Start of first RecursiveUnit in RBS address
    // After AddressingField array with 8-bit length fields
    RecursiveUnit = AddressingField + (N - 1) * 8;

    RemainLength = *Total_len - length(Rsv) - BIFT.entries;
    Index = GetFirstBitPosition(*BitStringA);

    while (Index) {
        PacketCopy = Copy(Packet);
        if (BIFT.BP[Index].recursive) {
            if (N == 1) {
                RecursiveUnitLength = RemainLength;
            } else {
                RecursiveUnitLength = *AddressingField;
                N--; 
                AddressingField += 8;
                RemainLength -= RecursiveUnitLength;
                RemainLength -= 8; // 8 bit of AddressingField
            }
            RewriteRBS(PacketCopy, RecursiveUnit, RecursiveUnitLength);
            SendTo(PacketCopy, BIFT.BP[Index].adjacency);
            RecursiveUnit += RecursiveUnitLength;
        } else {
            DisposeRBSheader(PacketCopy);
            SendTo(PacketCopy, BIFT.BP[Index].adjacency);
        }
        Index = GetNextBitPosition(*BitStringA, Index);
    }
}
```

Other thoughts

• Natural variable length address (structure)
  > Compared to naturally fixed-size in BIER/BIER-TE

• Forwarding plane complexity
  > Basic Bitstring replication the same as BIER-TE (only simple subset of adjacencies required)
  > Main added work:
    - For each R)eplicating adjacency:
      locate offset/length of Recursive Unit for adjacency, rewrite RBS to that Recursive Unit

• TBD: stochastical analysis/comparison of efficiency (#copies), compared to BIER/BIER-TE
  > Wide comparison space…

• Packet encoding
  > Not purpose of this draft
  > Could just use RFC8296 and be considered ‘yet another BIFT mode’ (BIER, BIER-TE, BIER-RBS)
  > Or any other encap if more desirable / applicable
THE END