Recursive Tree Structure (RTS) forwarding

PIM-WG, IETF119 Brisbane, v1.0

draft-eckert-pim-rts-forwarding-01

Toerless Eckert, Futurewei USA (tte@cs.fau.de) (Editor)
Michael Menth, (menth@uni-tuebingen.de),
Steffen Lindner, (steffen.lindner@uni-tuebingen.de)

Why are we here ?

BIER in large networks

BIER: (long) global bitstring in packet

Need to subdivide large network into groups of nodes (BFER) that share a bitstring (SI, SD)

Large networks that need to have N bitstring groups may need to send (worst case) N packets to reach N receivers

Each receiver is in a separate group == different SI or SD == separate packet

Admin/Ops/Deployment problem. defining/redefining those groups for large networks

Example: large DC 100,000 nodes. Customer VLANs with 10..100 nodes. Which may move between Server hosts due to VM migration.

Tree Engineering (BIER-TE)

Same scaling/admin problem, just much worse, because now we also need bits for all transit nodes.

Solution: RBS, RTS:

Packet header encodes actual desired delivery tree to receivers

Goal: Eliminate SI, SD static group definitions -

Gives tree engineering for free. Aka: will use explicit tree in header even if tree is just shortest-path-tree

Solution: Encode the actual (sub) tree in the header

1. Each router only need list of neighbors it needs to replicate to:

   Rx -> {R2, R4} R2, R4 = List of SIDs.

2. Each of those neighbors has their own sub-tree (unless it is a leaf on the tree). We call this sub-tree a Recursive Unit (RU):

   Rx -> { R2 -> RU2 , R4 -> RU4}

   For Rx, the RU for R2 and R4 are just opaque blobs of data. There is no parsing of them on Rx.

3. When replicating the packet to a neighbor, it only needs to receive the RU designated for it:

   RU2 -> { R5, R6 }

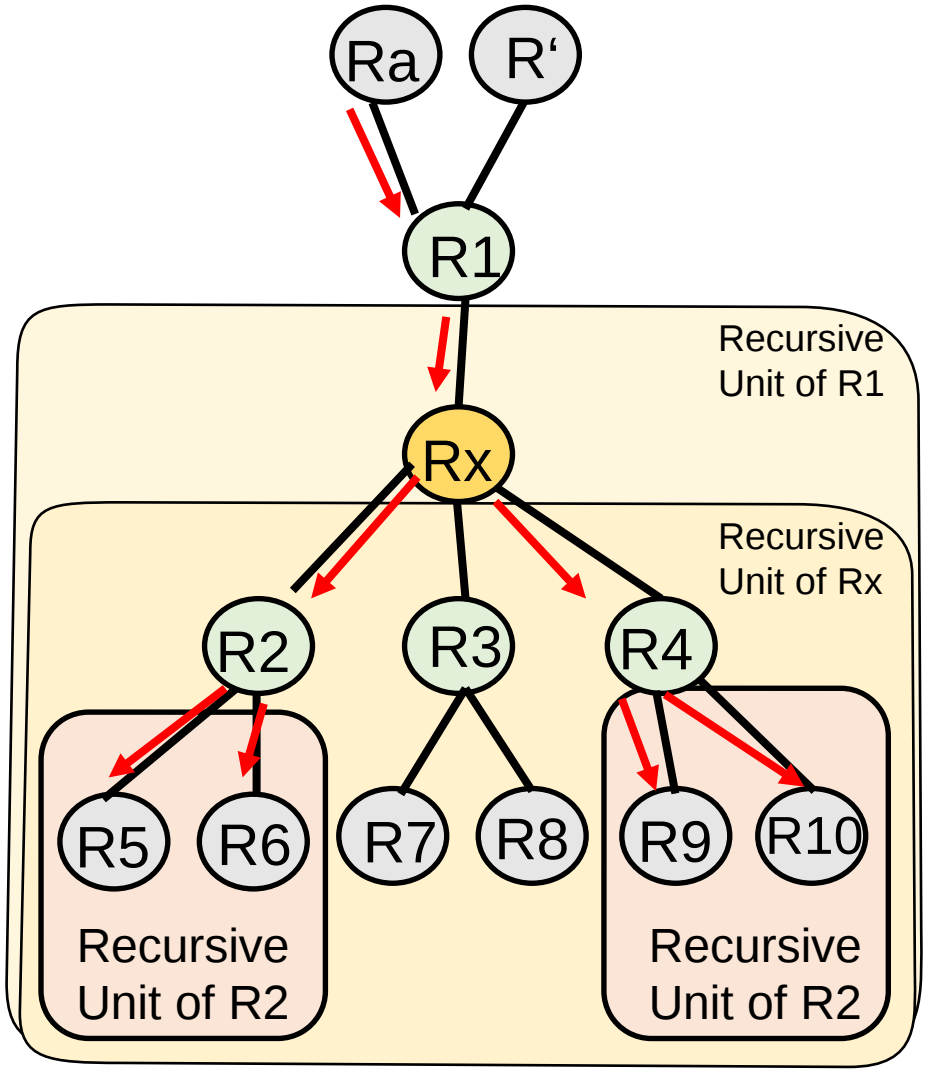
   RU4 -> { R9, R10 }

4. Packet from Ra -> R1 -> Rx:

   Ra -> {R1 -> {Rx -> {R2 -> { R5, R6 } , R4 -> { R9, R10 }}}}

   R1 -> {Rx -> {R2 -> { R5, R6 } , R4 -> { R9, R10 }}}

   Rx -> { R2 -> {R5, R6} , R4 -> {R9, R10}}



Tree encoded from Recursive Units (RU)

Tells one node what to do:

- “local bitstring” – for direct L2 neighbors. 16 neighbors == 16 bit bitstring
- “list of local SID” - 16 neighbors = list of 1 .. 16 * e.g.: 4-bit long SID.

  Example: For up to 4 neighbors this could be equal/shorter than bitstring
- “list of global SID” – list of SIDs long enough to identify all nodes in network: 16 ? 20 ? 24 bit ?

  Great when we do not need to steer packet across each hop (TE), but only get it to some remote nodes using shortest path to them.

  Not well feasible with bitstring!

Most efficient encoding ?

- Few initial nodes in tree using global SID list
- nodes towards leaves of tree use local SID or bitstring – depending on what is more compact.
- The bigger the tree, the more beneficial local bitstrings become

RTS for IETF118 (00) -> IETF119 (01)

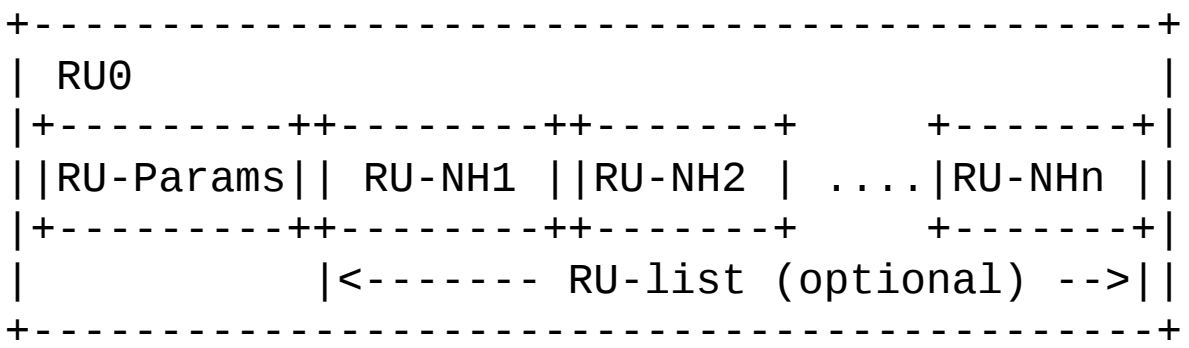
IETF118:

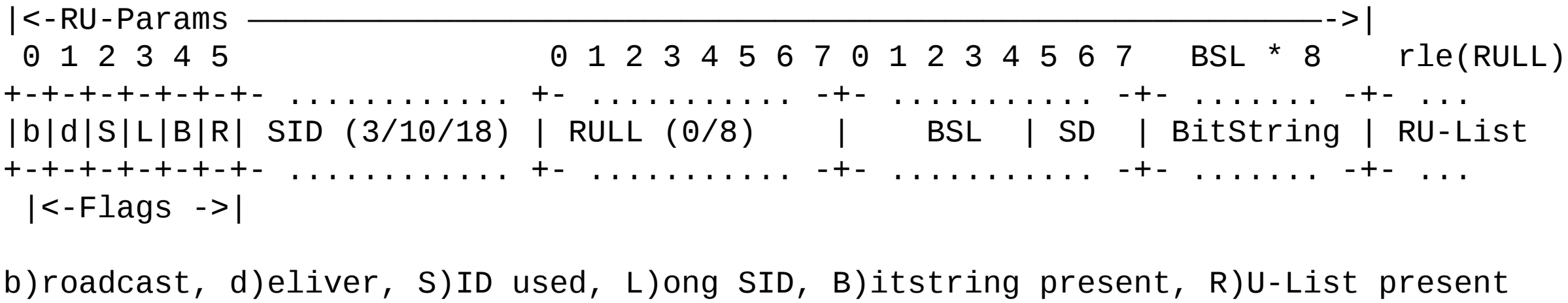
- P4 reference implementation separately supporting
  - Global SID trees only (“SEET”)
  - Simplified RBS (“local bitstrings only”)
- Each packet can only have a SID-tree or a Local-BitString-Tree

Since then work/P4 implementation see how it can improve

- SEET+: Packet can is SID tree with optional leaf-node Local Bitstrings
  - Limitations set by P4 reference platform (described in BIER0-WG)
- RTS-01 generalization:
  - Each node can have SID-list or Local BitString
  - But limited functionality nodes may not need to support both
    - This allows to work like SEET+, but be more flexible across nodes with more flexible forwarding plane

Encoding





b)roadcast, d)eliver, S)ID used, L)ong SID, B)itstring present, R)U-List present

Where to go from here

- Research paper
- Reference code useable by others
- Biggest opportunity ? Large-scale parallel AI in DC
  - Use of multicast still being researched / questioned
  - Fixed, well-known topologies – Spine-Leaf/Dragonfly/...
  - Research competition exist, but not single one for different topologies...compare
- Q: „safe“ vs. „most compact“ encoding
  - E.g.: do not include length fields when using known fixed topologies
    - Discuss point with Ice – e.g. when using SPine/Leaf
    - Signal length fields through control plane instead of header
- What is the target hardware ?
  - Differences in ASIC flexibility in Deep vs. Shallow buffer DC ASICs
  - DC-AI usecase requires ability to fit („cheap“) shallow buffer ASICs
  - Our reference P4 code should be good indication this is feasible

Questions ?

Encoding optimization for large trees

IPTV and other “broadcast” applications may need to
Go to many/most receivers (large tree).

Encoding optimization: “broadcast to all your leaf neighbors”

Simulation comparison BIER, Recursive Bitstrings for large SP
Number of copies even goes down with larger trees!