Trill Parent node shift, Mitigation. IETF 98, Chicago.

https://datatracker.ietf.org/doc/draft-rp-trill-parent-selection-03

R. Parameswaran,

Brocade Communications Inc.,

parameswaran.r7@gmail.com.

Problem Statement/Summary

- The parent selection rules, standardized in Trill's tree construction process, can lead to un-necessary shifts in parent-child relationships, in some situations.
- Aware of this impacting latency requirements for some customers.
- The draft presented here proposes two distinct solutions which can be used to address the problem.

Problem statement

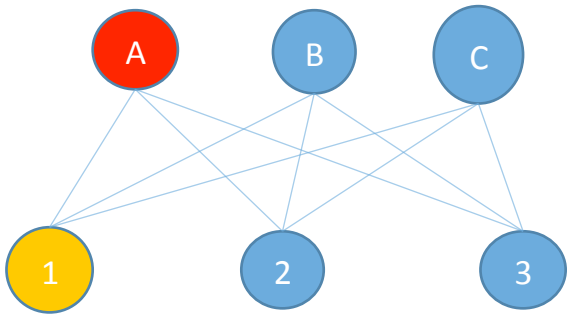
- What is the issue with Trill's standard tree construction method?
  - Let's see how Trill defines parent selection during tree construction.
    [RFC6325]:
    - "When building the tree number j, remember all possible equal cost parents for node N. After calculating the entire 'tree' (actually, directed graph), for each node N, if N has 'p' parents, then order the parents in ascending order according to the 7-octet IS-IS ID considered as an unsigned integer, and number them starting at zero. For tree j, choose N's parent as choice j mod p."

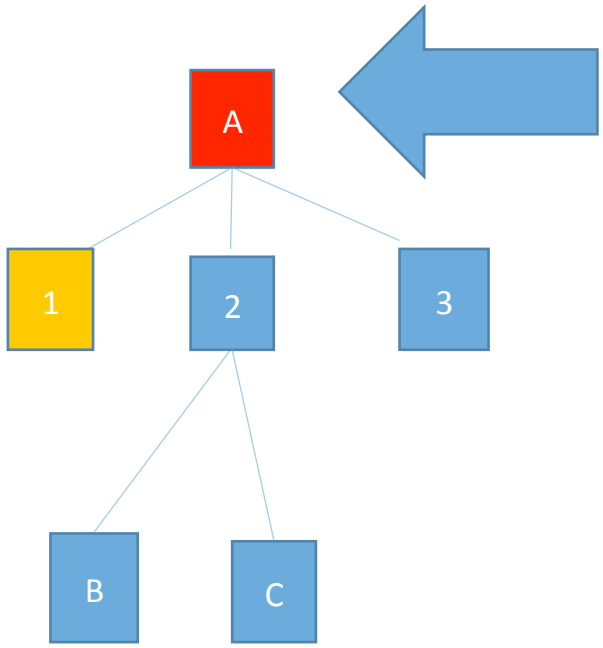
Problem statement (contd).

- There is an additional correction posted to this in [RFC7780]:
- [RFC7780], Section 3.4:

  “This is changed so that the selected parent MUST be (j-1) mod p. As a result, in the case above, tree 1 will select parent 0, and tree 2 will select parent 1. This change is not backward compatible with[RFC6325]. If all RBridges in a campus do not determine distribution trees in the same way, then for most topologies, the RPFC will drop many multi-destination packets before they have been properly delivered."

Problem Statement (Depiction)



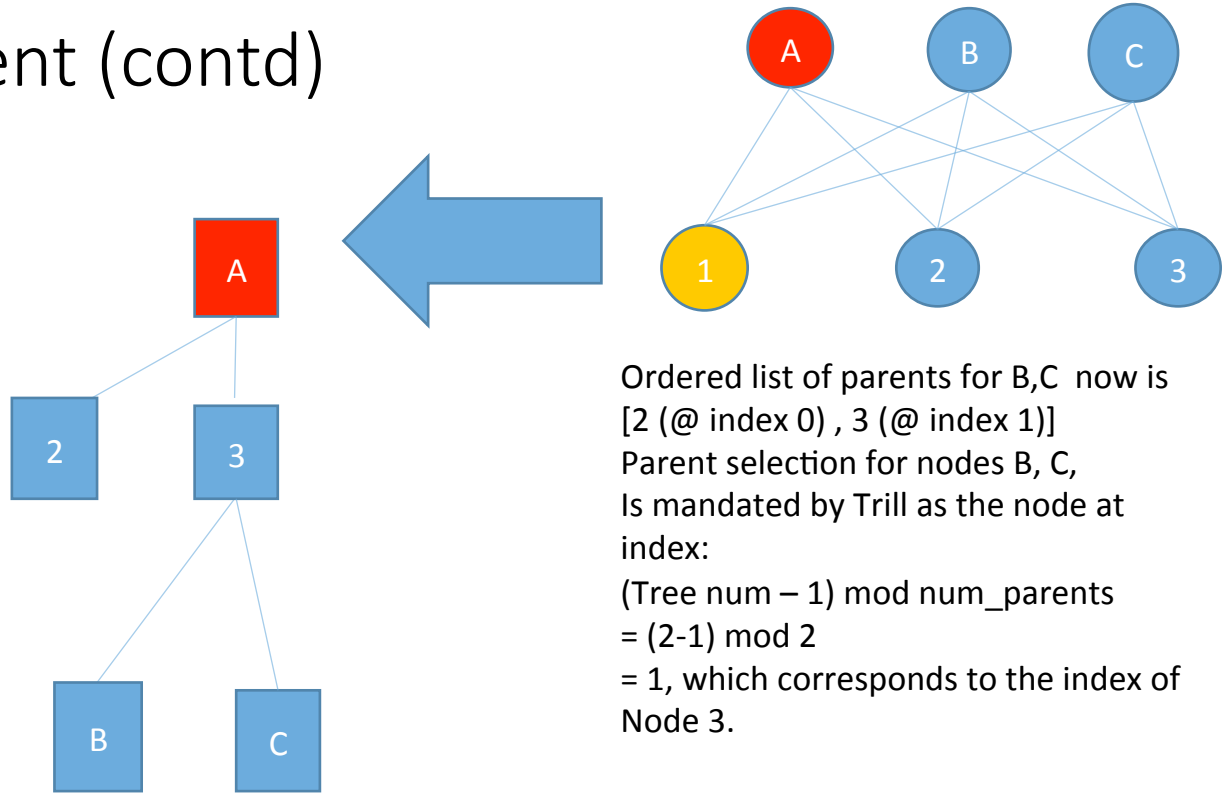


Consider **tree 2**, and say **node A** is the tree **root**. Now, consider what happens if Node 1 goes down?

Before Node 1 goes down the Ordered list is:
[1 (@ index 0), 2 (@ index 1) , 3]
Parent selection for nodes B, C,
Is mandated by Trill as the node at index:
(Tree num – 1) mod num_parents
= (2-1) mod 3
= 1, which corresponds to the index of Node 2.

Assume that nodes 1, 2, 3 are in sorted order, sorted by ascending order of ISIS 7 octet ID.

Problem Statement (contd)



Consider what happens if
Node 1
goes down?

Ordered list of parents for B,C now is
[2 (@ index 0) , 3 (@ index 1)]
Parent selection for nodes B, C,
Is mandated by Trill as the node at index:
(Tree num – 1) mod num_parents
= (2-1) mod 2
= 1, which corresponds to the index of Node 3.

B and C's parent shifted from node 2 to node 3 – this is unnecessary, since 2 never went down. Similar problems can happen with other tree numbers, and this can happen at each parent/child relationship in the tree.

Solution (Approach 1).

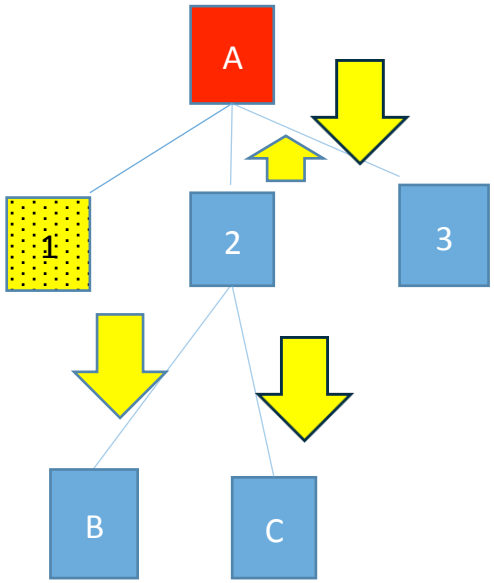
- How can this be solved?
- Affinity sub-TLV (It's a sub-TLV of the Router capability TLV).
- Other drafts/RFCs now use affinity sub-TLV in other scenarios.
- Affinity TLV basically dictates parent-child mappings.
- Is published by the parent, identifying the list of children it wants to bind to, and the specific tree on which this is to be done.
- Powerful sub-TLV which needs to be used within certain guide-lines.
- Applicability to this case, operator pins stickiness to children on a specific (parent) node, using a CLI. Operator takes responsibility of configuring CLI on only one of a set of possible parent nodes (should not be configured on more than one sibling, and should not be configured on the root of the tree).

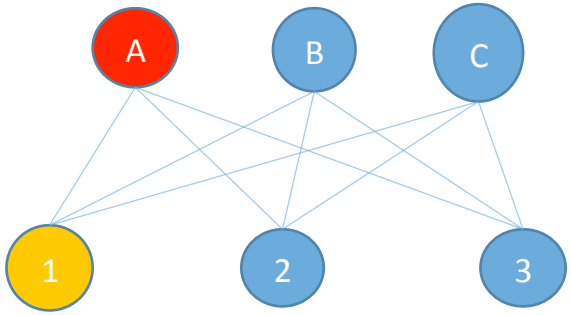
Solution (Approach 1) Contd.

- Operator configures parent stickiness on a particular parent (designated parent), for particular tree number.
- The designated parent runs through a tree calculation, ignores the default Trill parent selection rule, and asserts its right to be a parent, if during tree computation it finds itself to be a potential parent of one or more child nodes.
- Once the tree computation completes (with an additional stabilization timer), the designated parent node publishes an Affinity sub-TLV, identifying the child nodes and the tree number.
- This is also repeated in any subsequent tree computations.
- Other nodes in the network blindly honor the affinity sub-TLV sent by the designated parent, if any.
- Note that tree structure will change as links and nodes go down or come up in the network. The designated parent will either publish unchanged, or change, or retract affinity sub-TLV as network events change the tree, depending on whether it has children or not in the new tree order (designated parent makes a best-effort to try and preserve its existing child relationships, tree structure permitting, disregarding the default Trill tree construction rule, and ignoring its own affinity TLV in its own tree calculation).
- In the event of a retraction of the affinity sub-TLV, other nodes in the network fall back to the default Trill tree construction rules.

Solution, Approach 1:

Affinity sub-TLV, Tree = 2 , Child = B,C (Originated by 2) after operator config, before Node 1 goes down.





- Once 2 publishes an affinity sub-TLV, all other nodes in the network factor it, in their tree construction, using Node 2 as the parent for B,C in tree 2's construction.
- Node 2 does not blindly honor Trill's default parent selection rules (and ignores its own affinity sub-TLV) and instead, tries to assert/preserve its parent relationship to its children to the extent possible, and publishes or updates the affinity sub-TLV after its own tree construction of tree 2.

Consider what happens if Node 1 goes down? Node 2 publishes an affinity sub-TLV before Node 1 goes down, preventing the problem of B and C's parent-shift.

Solution (Approach 2).

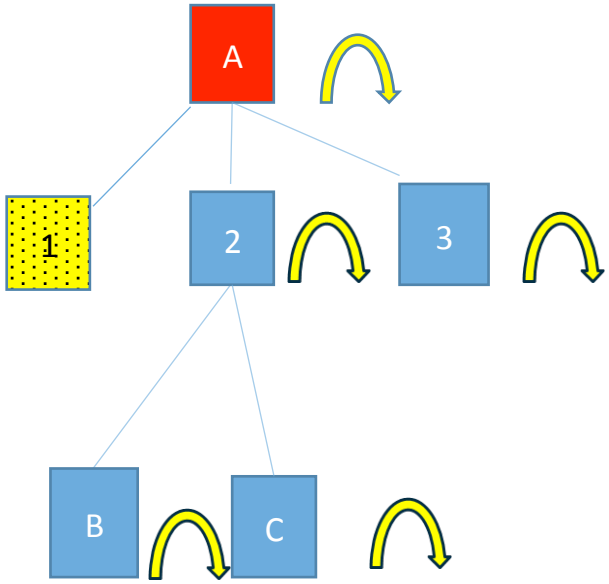
- Use a modified version of SPF which inserts a policy driven selector for the choice of parent when multiple parents can pull a child node into the SPF tree at the same optimal cost.
- Make the policy function choose based on a previous stable snap-shot of the same tree.
- Hence for a given child node, it will pick the same parent that it had in the previous stable snap-shot of the tree, before the network churn event happened. The very first tree calculation uses the default Trill parent selection rules.
- This determination happens in a distributed fashion at each node in the network.
- Hence, nodes in the network had better agree on what the previous stable snap-shot of the tree looked like.

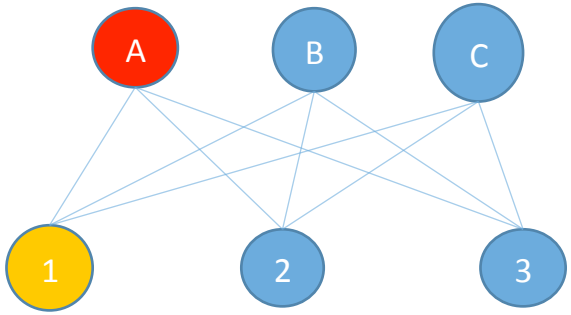
Solution (Approach 2) Contd:

- This is relatively difficult to do, but we can leverage the fact that by the time routes/result of the tree computation was downloaded to the RIB, all the nodes in the network agreed on the previous stable version of the tree, so use this as trigger to collect the snapshot.
- May need additional dampening, at the RIB trigger.
- Works best in small to mid-size networks.
- Special handling needed for link flaps, other events where the parent child relationship inverts etc..

Solution, Approach 2

Nodes latch to their previous stable tree computation and use that to guide tree construction.
A is tree root, Node 1 went down.





Consider tree 2, and say node A is the tree root. Now, consider what happens if Node 1 goes down? During tree calculation, because the prior tree calculation used Node 2 as the parent, the policy selection step continues to select Node 2 as the parent in new tree calculations.

Initial tree computation uses the Trill default rules. Subsequent tree computations use the previous stable snap-shot to drive parent selection.

Other Considerations:

- Between the two approaches, approach A is preferable, since affinity sub-TLV makes the network behavior more predictable, unless there are IPR considerations.
- No IPR on either of these approaches at Brocade Communications, Inc.

Status/Advancement:

- Requesting adoption as WG document, aware of customers looking for a way to prevent unnecessary parent shifts.
- A few logical inconsistencies in approach B, clarification in Approach A, fixed in a private copy of the draft, will upload after IETF98.
- Might remove or tweak approach B from the draft in subsequent uploads.
- Approach B as proposed here may not be feasible and/or can be tweaked , but the policy driven SPF tree computation proposed in approach B might have some value if there is interest in pursuing network wide alternative default parent selection rules.
- Approach A can be enhanced to add node redundancy for the sticky/designated parent.
- Planning to change the draft to Informational, assuming approach A remains feasible.