TRILL WG R. Parameswaran,
INTERNET-DRAFT Brocade Communications, Inc.
Intended Status:Informational April 22, 2017
Expires: October 20, 2017

TRILL: Parent node Shifts in Tree Construction, Mitigation.
<draft-rp-trill-parent-selection-03.txt>

Abstract

This draft documents a known problem in the TRILL tree construction mechanism and offers an approach requiring no change to the TRILL protocol in order to solve the problem.

Terminology and Acronyms.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Table of Contents

1. Introduction.

TRILL is a data center technology that uses link-state routing mechanisms in a layer 2 setting, and serves as a replacement for spanning-tree. TRILL uses trees rooted at pre-determined nodes as a way to distribute multi-destination traffic. Multi-destination traffic includes traffic such as layer-2 broadcast frames, unknown unicast flood frames, and layer 2 traffic with multicast MAC addresses (collectively referred to as BUM traffic). Multi-destination traffic is typically hashed onto one of the available trees and sent over the tree, potentially reaching all nodes in the network (hosts behind which may own/need the packet in question).

2. Tree construction in TRILL.

Tree construction in TRILL is defined by [RFC6325], with additional corrections defined in [RFC7780].

The tree construction mechanism used in TRILL codifies certain tree construction steps which make the resultant trees very brittle. Specifically, the parent selection mechanism in TRILL causes problems in case of node failures. TRILL uses the following rule - when constructing an SPF tree, if there are multiple possible parents for a given node (i.e. if multiple upstream nodes can potentially pull in a given node during SPF, all at the same cumulative cost, then the parent selection is imposed in the following manner):

[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."

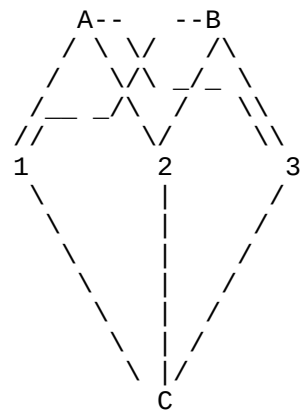
There is an additional correction posted to this in [RFC7780]:

[RFC7780], Section 3.4:

> "Section 4.5.1 of [RFC6325] specifies that, when building distribution tree number j, node (RBridge) N that has multiple possible parents in the tree is attached to possible parent number j mod p. Trees are numbered starting with 1, but possible parents are numbered starting with 0. As a result, if there are two trees and two possible parents, then in tree 1 parent 1 will be selected, and in tree 2 parent 0 will be selected.
>
> 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."

3. Issues with the TRILL tree construction algorithm.

With this tree construction mechanism in mind,let's look at the Spine-Leaf topology presented below and consider the calculation of Tree number 2 in TRILL. Assume all the links in the tree are at the same cost.



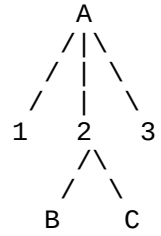
Assume that in the above topology, when ordered by 7-octet ISIS-id,

1 < 2 < 3 holds and that the root for Tree number 2 is A. Given the ordered set {1, 2, 3} , these nodes have the following indices (with a starting index of 0):

Node	Index
1	0
2	1
3	2

Given the SPF constraint and that the tree root is A, the parent for nodes 1,2, and 3 will be A. However, when the SPF algorithm tries to pull B or C into the tree, we have a choice of parents, namely 1, 2, or 3.

Given that this is tree 2, the parent will be the one with index (2-1) mod 3 (which is equal to 1). Hence the parent for node B will be node 2.

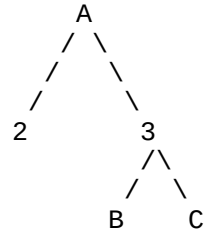


However, due to TRILL's parent selection algorithm, the sub-tree rooted at Node 2 will be impacted even if Node 1 or Node 3 go down.

Take the case where Node 1 goes down. Tree 2 must now be re-computed (this is normal) - but now, when the SPF computation is underway, when the SPF process tries to pull in B, the list of potential parents for B now are {2 and 3}. So, after ordering these by ISIS-Id as {2, 3} (where 2 is considered to be at index of 0 and 3 is considered to be at index 1), for tree 1, we apply TRILL's formula of:

Parent's index = (TreeNumber-1) mod Number_of_parents.
= (2-1) mod 2
= 1 mod 2
= 1 (which is the index of Node 3)

The re-calculated tree now looks as shown below. The shift in parent nodes (for B) may cause disruption to live traffic in the network, and is unnecessary in absolute terms because the existing parent for node B, node 2, was not perturbed in any way.



Aside from the disruption posed by the change in the tree links, depending upon how the concerned rbridges stripe vlans/FGLs across trees and how they may prune these, additional disruption is possible if the forwarding state on the new parent rbridge is not primed to match the new tree structure. This churn could simply be avoided with a better approach.

The parent shift issue noted above can be solved by using

the Affinity sub-TLV.

While the technique identified in this draft has an immediate benefit when applied to spine/leaf networks popular in data-center designs, nothing in the approach outlined below assumes a spine-leaf network. The technique presented below will work on any connected graph. Furthermore, no directional symmetry in link-cost is assumed.

4. Solution using the Affinity sub-TLV.

At a high level, this problem can be solved by having the affected parent send out an Affinity sub-TLV identifying the children for which it wants to preserve the parent-child relationship, subject to network events which may change the structure of the tree. The affected parent node would send out an Affinity sub-TLV with multiple Affinity records, one per child node, listing the concerned tree number.

It would be sufficient to have a local configuration option (e.g. a CLI) at one of the nodes which is deemed to be the parent of choice (referred to as designated parent below). The following steps provide a way to implement this proposal:

a. The operator locally configures the designated parent to indicate its stickiness in tree construction for a specific tree number and tree root via the Affinity sub-TLV. This can be done before tree construction if the operator consults the 7 octet ISIS-ID relative ordering of the concerned nodes and decides up-front which of the potential parent nodes should become the parent node for a given set of children on that tree number under the TRILL tree construction mechanism. The operator MUST configure the designated parent stickiness on only one node amongst a set of sibling (potential parent) nodes relative to the tree root for that tree number. It is suggested that the parent stickiness be configured on the node that would have been selected as the parent under default Trill parent selection rules. Parent stickiness MUST NOT be configured on the root of the tree, or if configured previously on a non-root node with the root for that tree shifting to that node subsequently, such configuration MUST be ignored on the root node.

b. On any subsequent SPF calculation after the operator configures the designated parent as indicated above, when the designated parent node finds that it could be a potential parent for one or more child nodes during tree construction, it declares itself to be the parent for the concerned child nodes, over-riding the default TRILL parent selection rules. The configured node advertises its parent preference via the Affinity sub-TLV when it completes a tree calculation, and finds itself the parent of one or more child nodes per the SPF tree calculation. The Affinity sub-TLV MUST reflect the appropriate tree number and the child nodes for which the concerned node is a parent node. The Affinity sub-TLV SHOULD be published when the tree computation is deemed to have converged (more on this under d. below).

c. Likewise, when any change event happens in the network, one which forces a tree re-calculation for the concerned tree, the designated parent node should run through the normal TRILL tree calculation agnostic of the fact that it has published an Affinity sub-TLV as well as agnostic of the default TRILL tree selection rules i.e the node asserts its right to be a parent without directly referencing either the default Trill parent selection rules or its own published Affinity sub-TLV in establishing parent relationships.

d. During the SPF tree calculation, the designated parent node should react in the following manner:

i. If the node is a potential parent for some of the children identified in an existing Affinity sub-TLV, if any, after convergence of the tree computation, the node MUST send out an (updated) Affinity sub-TLV identifying the correct sub-set of children for which the node aspires to establish/continue the parent relationship. This case would also apply if there are new child nodes for which the node is now a parent (however, see the conflicted Affinity sub-TLV rules in vii and j. below).

   For its own tree computation, the designated parent node MUST use itself as parent in order to pull the set of children identified during the SPF run into the tree, barring a conflicting affinity sub-TLV seen from another node (see vii. below for handling this case).

ii. If the tree structure changes such that the designated node is no longer a potential parent for any of the child nodes in the advertised Affinity sub-TLV, then it SHOULD retract the Affinity sub-TLV, upon convergence of the tree computation. In this case, the default TRILL tie-break rule would need to be used during SPF construction for the nodes that were children of this designated node previously. One specific case may be worth high-lighting - if a parent-child relationship inverts i.e. if the designated parent becomes a child of its former child node due to a change in the tree structure, it MUST exclude that child from its Affinity sub-TLV. In such case, if the designated parent node cannot maintain a parent relationship with any of its prior child nodes, then it MUST retract any previously published affinity sub-TLV.

iii. Nodes SHOULD use a convergence timer to track completion of the tree computation. If there are any additional tree computations while the convergence timer is running, the timer SHOULD be re-started/extended in order to absorb the interim network events. It is possible that the intended action at the expiration of the timer may change meanwhile. The timer needs to be large enough to absorb multiple network events that may happen due to a change in the physical state of the network, and yet short enough to avoid delaying the update of the Affinity sub-TLV.

iv. At the expiration of the convergence timer, the existing state of the tree MUST be compared with the existing Affinity sub-TLV and the intended change in the status of the Affinity sub-TLV is carried out e.g. a fresh publication, or an update to the list of children, or a retraction.

v. Alternately, the above steps (re-examination of the Affinity sub-TLV and update) MAY be tied to/triggered from the download of the tree routes to the L2 RIB, since that typically happens upon a successful computation of the complete tree. An additional stabilization timer could be used to counteract back-to-back L2 RIB downloads due to repeated computations of the tree due to a burst of network events.

vi. Note that this approach may cause an additional tree computation at remote nodes once the updated Affinity sub-TLV (or lack of it) is received/perceived, beyond the network events which led up to the change in the tree. In the case where an operator introduced a designated parent configuration on an existing tree, then remote nodes would need to receive the Affinity sub-TLV indicating the designated parent’s Affinity for its children before the remote nodes shift away from the default TRILL parent selection rules. However, in most cases, in steady state, this mechanism should cause very little tree churn unless

a designated parent configuration was introduced, removed, or a link between the designated parent and its children changed state. In cases where the network change event originated on the designated parent node, it may be possible to optimize on the churn by packing both the data bearing the network change event and the Affinity sub-TLV into the same link-state update packet.

vii. In situations where the designated parent node would normally originate an affinity sub-TLV to indicate affinity to a specific set of child nodes, it MUST NOT originate an Affinity sub-TLV if it sees an Affinity sub-TLV from some other node for the same tree number and for all of the same child-nodes, such that the other node's Affinity sub-TLV would win using the conflict tie-break rules in section 5.3 of [RFC7783]. Any existing Affinity sub-TLV already published by this node in such a situation MUST be retracted. If only some of the child nodes overlap between the two conflicting Affinity sub-TLVs, then this designated parent node MAY continue to publish its affinity sub-TLV listing its child nodes that are not in conflict with the other Affinity sub-TLV. Other guide-lines listed in [RFC7783] MUST be adhered to as well - the originator of the Affinity sub-TLV must name only directly adjacent nodes as children, and must not name the tree root as a child.

e. Situations where the node advertising the Affinity sub-TLV dies or restarts SHOULD be handled using the normal handling for such scenarios relating to the parent Router Capability TLV, and as specified in [RFC4971].

f. Situations where a parent-child link directly connected to the designated parent node constantly flaps, MUST be handled by having the designated parent node retract the Affinity sub-TLV, if it affects the parent-child relationships in consideration. The long-term state of the Affinity sub-TLV can be monitored by the designated parent node to see if it is being published and retracted repeatedly in multiple iterations or if a specific set of children are being constantly added and removed. The designated parent may resume publication of the Affinity sub-TLV once it perceives the network to be stable again in the future.

g. If the designated parent node is forced to retract its Affinity sub-TLV due to a change in the tree structure, it can then repeat these steps in a subsequent tree construction, if the same node becomes a parent again, so long as it perceives its parent-child links to be stable (free of link/node flaps).

h. In terms of nodes that do not support this draft, they are expected to seamlessly inter-operate with this draft, so long as they understand and honor the Affinity sub-TLV. The draft assumes that most TRILL implementations now support the Affinity sub-TLV. In any case, the guide-lines specified in section 4.1 of [RFC7783] MUST be used i.e. if all nodes in the network do not support the Affinity sub-TLV then the network must default to the Trill parent selection rules.

i. Remote nodes MUST default to the Trill parent selection rules if they do not see an Affinity sub-TLV sent by any node in the network.

j. At remote nodes, conflicting Affinity sub-TLVs from different originators for the same tree number and child node MUST be handled as specified in section 5.3 of [RFC7783], namely by selecting the Affinity sub-TLV originated by the node with the highest priority to be a tree root, with System-ID as tie-breaker.

5. Network wide selection of computation algorithm.

The proposed solution above does not need any operational change to the TRILL protocol, beyond the usage of the Affinity sub-TLV (which is already in the proposed standard) for the use case identified in this draft.

6. Relationship to draft-ietf-trill-resilient-trees.

Given that both draft-ietf-trill-resilient-trees, and draft-rp-trill-parent-selection-03 drafts use the Affinity sub-TLV, it is worthwhile to examine if there is any functional overlap between the two drafts. At a high level, the two drafts have different goals and appear to solve unrelated problems.

draft-ietf-trill-resilient-trees relates to link protection, and defines the notion of a primary distribution tree and a backup distribution tree (DT), where these trees are intentionally kept link disjoint to the extent possible, and the backup tree is pre-programmed in the hardware, and activated either up front or upon failure of the primary distribution tree.

On the other hand, draft-rp-trill-parent-selection-03 protects parent-child relationships of interest on the primary DT, and has no direct notion of a backup DT.

draft-ietf-trill-resilient-trees considers the following algorithmic approaches to the building the backup distribution tree (section numbers listed below are from draft-ietf-trill-resilient-trees):

1. Operator hand-configuration for links on the backup DT/manual generation of Affinity sub-TLV - this is very tedious and unlikely to scale or be implemented in practice, and hence is disregarded in the analysis here.

2. Section 3.2.1.1a: Use of MRT algorithms (which will produce conjugate trees - link disjoint trees with roots for primary and backup trees that are coincident on the same rBridge).

3. Section 3.2.1.1b: Once the primary DT is constructed, the links used in the primary DT are additively cost re-weighted, and a second SPF is run to derive the links comprising the backup DT. Affinity sub-TLV is used to mark links on the back-up DT which are not also on the primary DT. This approach can handle conjugate trees as well as non-conjugate trees (link disjoint trees that are rooted at different rBridges).

4. Section 3.2.2: A variation on the section 3.2.1.1b approach, but without Affinity sub-TLV advertisement. Once the primary DT is constructed, costs for links on the primary DT are multiplied by a fixed multiplier to prevent them from being selected in a subsequent SPF run, unless there is no other choice, and the subsequent SPF yields links on the backup DT.

All of the approaches above yield maximally link disjoint trees, when applied as prescribed.

Approach 4 above does not seem to use Affinity sub-TLVs and instead seems to depend upon a network wide agreement on the alternative tree computation algorithm being used.

Approaches 2 and 3 use Affinity sub-TLV on the backup DT, for links that are not already on the primary DT. The primary DT does not appear to use Affinity sub-TLVs. Additionally, from an end-to-end perspective the backup DT comes into picture when the primary DT fails (this is effectively true even in the 1+1 protection mechanism

and in the local protection case), and then again, only until the primary DT is recalculated. Once the primary DT is recalculated, the backup DT is recalculated as well, and can change corresponding to the new primary DT.

draft-ietf-trill-resilient-trees cannot directly prevent/mitigate a parent node shift on the primary DT at a given parent node, and while usage of the Affinity sub-TLV on the backup DT might confer a parent affinity on some nodes on the backup DT, these are not necessarily the nodes on which the network operator may want/prefer an explicit parent affinity. Further, the backup DT is only used on a transient basis, from a forwarding perspective, until the primary DT is recomputed.

However, a parent shift can be triggered by link or node failure. In a situation where both drafts are active in the implementation, failure of a specific link may cause the backup DT to kick in, but when the primary DT is re-calculated, draft-rp-trill-parent-selection-03 can be used to preserve parent-child relationships on the primary DT, to the extent possible, during the re-calculation. So, there does not appear to be a direct functional overlap in the simultaneous usage of these drafts, and it ought to be possible to use both drafts simultaneously, so long as the primary and back-up DTs can be uniquely identified/differentiated.

7. Security Considerations.

The proposal primarily influences tree construction and tries to preserve parent-child relationships in the tree from prior computations of the same tree, without changing any of operational aspects of the protocol. Hence, no new security considerations for TRILL are raised by this proposal.

8. IANA Considerations.

No new registry entries are requested to be assigned by IANA. The Affinity Sub-TLV has been defined in [RFC7176], and this proposal does not change its semantics in any way.

9. Informative References.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC6325] Perlman, R., Eastlake 3rd, D., Dutt, D., Gai, S., and A. Ghanwani, "Routing Bridges (RBridges): Base Protocol Specification", RFC 6325, DOI 10.17487/RFC6325, July 2011, <http://www.rfc-editor.org/info/rfc6325>.

[RFC7780] - Eastlake 3rd, D., Zhang, M., Perlman, R., Banerjee, A., Ghanwani, A., and S. Gupta, "Transparent Interconnection of Lots of Links (TRILL): Clarifications, Corrections, and Updates", RFC 7780, DOI 10.17487/RFC7780, February 2016, <http://www.rfc-editor.org/info/rfc7780>.

[RFC7783] Senevirathne, T., Pathangi, J., Hudson, J., "Coordinated Multicast Trees (CMT) for Transparent Interconnection of Lots of Links (TRILL)", RFC 7783, February 2016, <http://datatracker.ietf.org/doc/rfc7783>

[RFC4971] Vasseur, JP., Shen, N., Aggarwal, R., "Intermediate System to Intermediate System (IS-IS) Extensions for Advertising Router Information", RFC 4971, July 2007, <http://datatracker.ietf.org/doc/rfc4971>

Author’s Address:

R. Parameswaran,
Brocade Communications, Inc.
120 Holger Way,
San Jose, CA 95134.

Email: parameswaran.r7@gmail.com