

TRILL Working Group  
INTERNET-DRAFT  
Intended status: Proposed Standard  
Expires: August 19, 2018

R. Parameswaran  
Individual Contributor  
February 15, 2018

**TRILL (Transparent Interconnection of Lots of Links):  
Mitigation of Parent Node Shifts in Tree Construction**  
[<draft-ietf-trill-parent-selection-03.txt>](#)

Abstract

This document describes a known problem in the TRILL tree construction mechanism and offers an approach requiring no change to the TRILL protocol that solves the problem.

Status of This Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Distribution of this document is unlimited. Comments should be sent to the authors or the TRILL working group mailing list:  
[trill@ietf.org](mailto:trill@ietf.org).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/1id-abstracts.html>. The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.



## Table of Contents

<a href="#">1.</a>	Introduction.....	<a href="#">3</a>
<a href="#">1.1</a>	Terminology and Acronyms.....	<a href="#">3</a>
<a href="#">2.</a>	Tree construction in TRILL.....	<a href="#">3</a>
<a href="#">3.</a>	Issues with the TRILL tree construction algorithm.....	<a href="#">4</a>
<a href="#">4.</a>	Solution using the Affinity sub-TLV.....	<a href="#">6</a>
<a href="#">5.</a>	Network wide selection of computation algorithm.....	<a href="#">10</a>
<a href="#">6.</a>	Security Considerations.....	<a href="#">10</a>
<a href="#">7.</a>	IANA Considerations.....	<a href="#">10</a>
<a href="#">8.</a>	Normative References.....	<a href="#">11</a>
<a href="#">9.</a>	Informative References.....	<a href="#">11</a>
<a href="#">10.</a>	Acknowledgements.....	<a href="#">11</a>
	Author's Address:.....	<a href="#">12</a>



## **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 the spanning-tree protocol. TRILL uses Multi-destination trees rooted at predetermined nodes as a way to distribute multi-destination traffic.

Multi-destination traffic includes traffic such as layer-2 broadcast frames, unknown unicast flooded 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).

### **1.1 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](#)].

## **2. Tree construction in TRILL**

Tree construction in TRILL is defined by [[RFC6325](#)], with corrections defined in [[RFC7780](#)].

The tree construction mechanism used in TRILL codifies certain tree construction steps which make the resultant trees brittle as explained below. 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  $j \bmod p$ ."

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

[[RFC7780](#)], [Section 3.4](#):

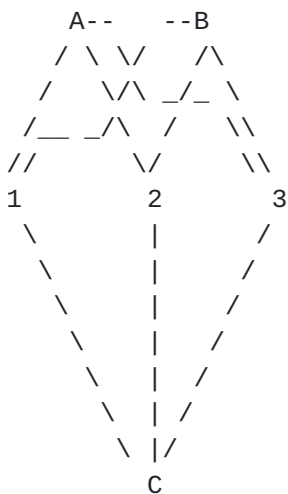


"[Section 4.5.1 of \[RFC6325\]](#) specifies that, when building distribution tree number  $j$ , node (RBridge)  $N$  that has multiple possible parents in the tree is attached to possible parent number  $j \bmod p$ . Trees are numbered starting with 1, but possible parents are numbered starting with 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  $(j-1) \bmod p$ . 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 the tree construction mechanism in [Section 2](#) 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 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):

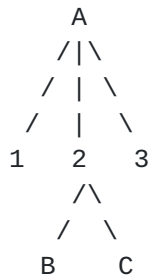
Node	Index
1	0
2	1
3	2





Given the SPF (Shortest Path First) 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  $(2-1) \bmod 3$  (which is equal to 1). Hence the parent for node B will be the node with an index value of 1, which is node 2.

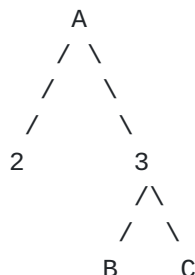


However, due to TRILL's parent selection algorithm, the sub-tree rooted at Node 2 will be impacted even if Node 1 or Node 3 go down.

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 =  $(\text{TreeNumber}-1) \bmod \text{Number\_of\_parents}$ .  
 $= (2-1) \bmod 2$   
 $= 1 \bmod 2$   
 $= 1$  (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 distribute 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 which is specified in [[RFC7176](#)].

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, despite network events which may change the structure of the tree. The concerned parent node would send out an Affinity sub-TLV with multiple Affinity records, one per child node, listing the affected tree number.

It would be sufficient to have a local RBridge configuration option at one of the nodes that is the parent chosen (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, overriding 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 MUST run through the normal TRILL tree calculation agnostic to the fact that it has published an Affinity sub-TLV and agnostic to the default TRILL tree selection rules i.e the node asserts its right to be a parent (based on its configuration as a designated parent) without directly referencing 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 i 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 later 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-breaking 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 highlighting - 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 guidelines 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 [\[RFC7981\]](#).
- 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. 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.
- i. 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.

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 guidelines specified in [section 4.1 of \[RFC7783\]](#) MUST be used i.e. if all nodes in the network do not announce support of the Affinity sub-TLV then the network MUST default to the TRILL parent selection rules.

## **6. 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 operational aspects of the protocol (this proposal does not introduce any new TLV/sub-TLV). Hence, no new security considerations for TRILL are raised by this proposal.

## **7. IANA Considerations**

This document requires no actions by IANA. The Affinity Sub-TLV has been defined in [\[RFC7176\]](#), and this proposal requires use of this Sub-TLV but does not change its semantics in any way.



## **8. Normative 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 (R Bridges): 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>>
- [RFC7981] Ginsberg, L., Previdi, S., Chen, M., "IS-IS Extensions for Advertising Router Information", [RFC 7981](#), October 2016, <<http://datatracker.ietf.org/doc/rfc7981>>
- [RFC7176] Eastlake 3'rd, D., et al, "Transparent Interconnection of Lots of Links (TRILL) Use of IS-IS", [RFC 7176](#), May 2014, <<http://datatracker.ietf.org/doc/rfc7176>>

## **9. Informative References**

None.

## **10. Acknowledgements**

I would like to thank Donald Eastlake for his help in preparing the current iteration of the draft, and for reviewing prior iterations.



## Author's Address:

Ramkumar Parameswaran,  
Individual contributor,  
PO Box 2788  
Cupertino, CA 95015.

Email: parameswaran.r7@gmail.com

## Copyright, Disclaimer, and Additional IPR Provisions

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

The definitive version of an IETF Document is that published by, or under the auspices of, the IETF. Versions of IETF Documents that are published by third parties, including those that are translated into other languages, should not be considered to be definitive versions of IETF Documents. The definitive version of these Legal Provisions is that published by, or under the auspices of, the IETF. Versions of these Legal Provisions that are published by third parties, including those that are translated into other languages, should not be considered to be definitive versions of these Legal Provisions. For the avoidance of doubt, each Contributor to the IETF Standards Process licenses each Contribution that he or she makes as part of the IETF Standards Process to the IETF Trust pursuant to the provisions of [RFC 5378](#). No language to the contrary, or terms, conditions or rights that differ from or are inconsistent with the rights and licenses granted under [RFC 5378](#), shall have any effect and shall be null and void, whether published or posted by such Contributor, or included with or in such Contribution.

