TRILL WG
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2018

Radia Perlman
Dell EMC
Fangwei Hu
ZTE Corporation
Donald Eastlake
Ting Liao
Huawei Technologies

Mar 11, 2018

TRILL Smart Endnodes
draft-ietf-trill-smart-endnodes-11.txt

Abstract

This draft addresses the problem of the size and freshness of the endnode learning table in edge RBridges, by allowing endnodes to volunteer for endnode learning and encapsulation/decapsulation. Such an endnode is known as a "Smart Endnode". Only the attached edge RBridge can distinguish a "Smart Endnode" from a "normal endnode". The Smart Endnode uses the nickname of the attached edge RBridge, so this solution does not consume extra nicknames. The solution also enables Fine Grained Label aware endnodes.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The IETF TRILL (Transparent Interconnection of Lots of Links) protocol [RFC6325] [RFC7780] provides optimal pair-wise data frame forwarding without configuration, safe forwarding even during periods of temporary loops, and support for multipathing of both unicast and multicast traffic. TRILL accomplishes this by using IS-IS [IS-IS] [RFC7176] link state routing and encapsulating traffic using a header that includes a hop count. Devices that implement TRILL are called "RBridges" (Routing Bridges) or "TRILL Switches".

An RBridge that attaches to endnodes is called an "edge RBridge" or "edge TRILL Switch", whereas one that exclusively forwards encapsulated frames is known as a "transit RBridge" or "transit TRILL Switch". An edge RBridge traditionally is the one that encapsulates a native Ethernet frame with a TRILL header, or that receives a TRILL-encapsulated packet and decapsulates the TRILL header. To encapsulate efficiently, the edge RBridge must keep an "endnode table" consisting of (MAC, Data Label, TRILL egress switch nickname) sets, for those remote MAC addresses in Data Labels currently communicating with endnodes to which the edge RBridge is attached.

These table entries might be configured, received from ESADI [RFC7357], looked up in a directory [RFC7067], or learned from decapsulating received traffic. If the edge RBridge has attached endnodes communicating with many remote endnodes, this table could become very large. Also, if one of the MAC addresses and Data Labels in the table has moved to a different remote TRILL switch, it might be difficult for the edge RBridge to notice this quickly, and because the edge RBridge is encapsulating to the incorrect egress RBridge, the traffic will get lost.

2. Conventions used in this document

2.1. Terminology

Edge RBridge: An RBridge providing endnode service on at least one of its ports. It is also called an edge TRILL Switch.

Data Label: VLAN or FGL.

DRB: Designated RBridge [RFC6325].

ESADI: End Station Address Distribution Information [RFC7357].

FGL: Fine Grained Label [RFC7172].

IS-IS: Intermediate System to Intermediate System [IS-IS].

PDU: Protocol Data Unit.

RBridge: Routing Bridge, an alternative name for a TRILL switch.

Smart Endnode: An endnode that has the capability specified in this document including learning and maintaining (MAC, Data Label, Nickname) entries and encapsulating/decapsulating TRILL frame.

Transit RBridge: An RBridge exclusively forwards encapsulated frames. It is also called a transit TRILL Switch.

TRILL: Transparent Interconnection of Lots of Links [RFC6325][RFC7780].

TRILL ES-IS: TRILL End System to Intermediate System, is a variation of TRILL IS-IS designed to operate on a TRILL link among and between one or more TRILL switches and end stations on that link[RFC8171].

TRILL Switch: a device that implements the TRILL protocol; an alternative term for an RBridge.

2.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Solution Overview

The Smart Endnode solution defined in this document addresses the problem of the size and freshness of the endnode learning table in edge RBridges. An endnode E, attached to an edge RBridge R, tells R that E would like to be a "Smart Endnode", which means that E will encapsulate and decapsulate the TRILL frame, using R’s nickname. Because E uses R’s nickname, this solution does not consume extra nicknames.

Take Figure 1 as the example Smart Endnode scenario: RB1, RB2 and RB3 are the RBridges in the TRILL domain, and SE1 and SE2 are the Smart Endnodes which can encapsulate and decapsulate the TRILL packets. RB1 is the edge RB that SE1 and SE2 have attached to. RB1 assigns one of its nicknames to be used by SE1 and SE2.

Each Smart Endnode, SE1 and SE2, uses RB1’s nickname when encapsulating, and maintains an endnode table of (MAC, label, TRILL egress switch nickname) for remote endnodes that it (SE1 or SE2) is corresponding with. RB1 does not decapsulate packets destined for SE1 or SE2, and does not learn (MAC, label, TRILL egress switch nickname) for endnodes corresponding with SE1 or SE2, but RB1 does decapsulate, and does learn (MAC, label, TRILL egress switch nickname) for any endnodes attached to RB1 that have not declared themselves to be Smart Endnodes.

Just as an RBridge learns and times out (MAC, label, TRILL egress switch nickname), Smart Endnodes SE1 and SE2 also learn and time out endnode entries. However, SE1 and SE2 might also determine, through ICMP messages or other techniques that an endnode entry is not successfully reaching the destination endnode, and can be deleted, even if the entry has not timed out.

If SE1 wishes to correspond with destination MAC D, and no endnode entry exists, SE1 will encapsulate the packet as an unknown destination, or consulting a directory [RFC7067] (just as an RBridge would do if there was no endnode entry).

```
+----------+
|SE1(Smart |
|Endnode1) |  \      +------------------------------+
+----------+   \    /                                \
                \  /+------+   +------+    +-----+    \   +-----------+
                /-+-| RB 1 |---|  RB2 |----| RB3 |-----+--|Endnode3   |
               /  | +------+   +------+    +-----+     |  |MAC=D      |
+----------+ /     \                                  /   +-----------+
|SE2(Smart |        \                                /
| Endnode2)|         +------------------------------+
+----------+
```

Figure 1 Smart Endnode Scenario

The mechanism in this draft is that the Smart Endnode SE1 issues a Smart-Hello, indicating SE1’s desire to act as a Smart Endnode, together with the set of MAC addresses and Data Labels that SE1 owns. The Smart-Hello is used to announce the Smart Endnode capability and parameters (such as MAC address, Data Label etc.). The Smart-Hello is a type of TRILL ES-IS PDU, which is specified in section 5 of [RFC8171]. The detailed content for a Smart Endnode’s Smart-Hello is defined in section 4.

If RB1 supports having a Smart Endnode neighbor it also sends Smart-Hellos. The Smart Endnode learns from RB1’s Smart-Hellos what RB1’s nickname is and which trees RB1 can use when RB1 ingresses multi-destination frames. Although Smart Endnode SE1 transmits Smart-Hellos, it does not transmit or receive LSPs or E-L1FS FS-LSPs [RFC7780].

Since a Smart Endnode can encapsulate TRILL Data packets, it can cause the Inner.Lable to be a Fine Grained Label [RFC7172], thus this method supports FGL aware endnodes. When and how a Smart Endnode decides to use the FGL instead of VLANs to encapsulate the TRILL Data packet is out of scope in this document.

4. Smart-Hello Mechanism between Smart Endnode and RBridge

The subsections below describe Smart-Hello messages.

4.1. Smart-Hello Encapsulation

Although a Smart Endnode is not an RBridge, does not send LSPs or maintain a copy of the link state database, and does not perform routing calculations, it is required to have a "Hello" mechanism (1) to announce to edge RBridges that it is a Smart Endnode and (2) to tell them what MAC addresses it is handling in what Data Labels. Similarly, an edge RBridge that supports Smart Endnodes needs a message (1) to announce that support, (2) to inform Smart Endnodes what nickname to use for ingress and what nickname(s) can be used as egress nickname in a multi-destination TRILL Data packet, and (3) the list of Smart Endnodes it knows about on that link.

The messages sent by Smart Endnodes and by edge RBridges that support Smart Endnodes are called "Smart-Hellos". The Smart-Hello is a type of TRILL ES-IS PDU, which is specified in [RFC8171].

The Smart-Hello Payload, both for Smart-Hellos sent by Smart Endnodes and for Smart-Hellos sent by Edge RBridges, consists of TRILL IS-IS TLVs as described in the following two sub-sections. The non-extended format is used so TLVs, sub-TLVs, and APPsub-TLVs have an 8-bit size and type field. Both types of Smart-Hellos MUST include a Smart-Parameters APPsub-TLV as follows inside a TRILL GENINFO TLV:

```
              +-+-+-+-+-+-+-+-+-
              |Smart-Parameters|                 (1 byte)
              +-+-+-+-+-+-+-+-+-
              |    Length      |                 (1 byte)
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
              |        Holding Time             |  (2 bytes)
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
              |             Flags               |  (2 bytes)
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2 Smart Parameters APPsub-TLV

o Type: APPsub-TLV type Smart-Parameters, value is TBD1.

o Length: 4.

o Holding Time: A time in seconds as an unsigned integer. It has the same meaning as the Holding Time field in IS-IS Hellos [IS-IS]. A Smart Endnode and an Edge RBridge supporting Smart Endnodes MUST send a Smart-Hello at least three times during their Holding Time. If no Smart-Hellos is received from a Smart Endnode or Edge RBridge within the most recent Holding Time it sent, it is assumed that it is no longer available.

o Flags: At this time all of the Flags are reserved and MUST be send as zero and ignored on receipt.

If more than one Smart Parameters APPsub-TLV appears in a Smart-Hello, the first one is used and any following ones are ignored. If no Smart Parameters APPsub-TLV appears in a Smart-Hello, that Smart-Hello is ignored.

4.2. Edge RBridge’s Smart-Hello

The edge RBridge’s Smart-Hello contains the following information in addition to the Smart-Parameters APPsub-TLV:

o RBridge’s nickname. The nickname sub-TLV, specified in section 2.3.2 in [RFC7176], is reused here carried inside a TLV 242 (IS-IS router capability) in a Smart-Hello frame. If more than one nickname appears in the Smart-Hello, the first one is used and the following ones are ignored.

o Trees that RB1 can use when ingressing multi-destination frames. The Tree Identifiers Sub-TLV, specified in section 2.3.4 in [RFC7176], is reused here.

o Smart Endnode neighbor list. The TRILL Neighbor TLV, specified in section 2.5 in [RFC7176], is reused for this purpose.

An Authentication TLV MAY also be included.

4.3. Smart Endnode’s Smart-Hello

A new APPsub-TLV (Smart-MAC TLV) is defined for use by Smart Endnodes as defined below. In addition, there will be a Smart-Parameters APPsub-TLV and there MAY be an Authentication TLV in a Smart Endnode Smart-Hello.

If there are several VLANs/FGL Data Labels for that Smart Endnode, the Smart-MAC APPsub-TLV is included several times in Smart Endnode’s Smart-Hello. This APPsub-TLV appears inside a TRILL GENINFO TLV.

```
   +-+-+-+-+-+-+-+-+
   |Type=Smart-MAC |                          (1 byte)
   +-+-+-+-+-+-+-+-+
   |   Length      |                          (1 byte)
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |F|M|   RSV     |  VLAN/FGL Data Label  |  (4 bytes)
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          MAC (1)      (6 bytes)                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      .................                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          MAC (N)      (6 bytes)                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3 Smart-MAC APPsub-TLV

o Type: TRILL APPsub-TLV Type Smart-MAC, value is TBD2.

o Length: Total number of bytes contained in the value field of the TLV, that is, the sum of the length of the F/M/RSV/FGL Data Label fields and 6 times the number of MAC addresses present. So, if there are n MAC addresses, this is 4+6*n.

o F: 1 bit. If it is set to 1, it indicates that the endnode supports FGL data labels [RFC7172], and that this Smart-MAC APPsub-TLV has an FGL in the following VLAN/FGL field. Otherwise, the VLAN/ FGL Data Label field is a VLAN ID.(See below for the format of the VLAN/FGL Data Label field).

o M: 1 bit. If it is set to 1, it indicates multi-homing(See Section 6). If it is set to 0, it indicates that the Smart Endnodes are not using multi-homing.

o RSV: 6 bits, is reserved for the future use.

o VLAN/FGL Data Label: 24bits. If F is 1, this field is a 24-bit FGL Data Label for all subsequent MAC addresses in this APPsub-TLV. Otherwise, if F is 0, the lower 12 bits is the VLAN of all subsequent MAC addresses in this APPsub-TLV, and the upper 12 bits is not used(sent as zero and ignored on receipt). If there is no VLAN/FGL data label specified, the VLAN/FGL Data Label is zero.

o MAC(i): This is a 48-bit MAC address reachable in the Data Label sent by the Smart Endnode that is announcing this APPsub-TLV.

5. Data Packet Processing

The subsections below specify Smart Endnode data packet processing. All TRILL Data packets sent to or from Smart Endnodes are sent in the Designated VLAN [RFC6325] of the local link but do not necessarily have to be VLAN tagged.

5.1. Data Packet Processing for Smart Endnode

A Smart Endnode does not issue or receive LSPs or E-L1FS FS-LSPs or calculate topology. It does the following:

- A Smart Endnode maintains an endnode table of (the MAC address of remote endnode, Data Label, the nickname of the edge RBridge’s attached) entries of end nodes with which the Smart Endnode is communicating. Entries in this table are populated the same way that an edge RBridge populates the entries in its table:

  - learning from (source MAC address ingress nickname) on packets it decapsulates.

  - by querying a directory [RFC7067].

  - by having some entries configured.

- When Smart Endnode SE1 wishes to send unicast frame to remote node D, if (MAC address of remote endnode D, Data Label, nickname) entry is in SE1’s endnode table, SE1 encapsulates the ingress nickname as the nickname of the RBridge(RB1), egress nickname as indicated in D’s table entry. If D is unknown, SE1 either queries a directory or encapsulates the packet as a multi-destination frame, using one of the trees that RB1 has specified in RB1’s Smart-Hello. The mechanism for querying a directory is given in [RFC8171].

- When SE1 wishes to send a BUM packet to the TRILL campus, SE1 encapsulates the packet using one of the trees that RB1 has specified.

If the Smart Endnode SE1 sends a multi-destination TRILL Data packet, the destination MAC of the outer Ethernet is the All-RBridges multicast address.

The Smart Endnode SE1 need not send Smart-Hellos as frequently as normal RBridges. These Smart-Hellos could be periodically unicast to the Appointed Forwarder RB1. In case RB1 crashes and restarts, or the DRB changes and SE1 receives the Smart-Hello without mentioning SE1, SE1 SHOULD send a Smart-Hello immediately. If RB1 is Appointed

Forwarder for any of the VLANs that SE1 claims, RB1 MUST list SE1 in its Smart-Hellos as a Smart Endnode neighbor.

5.2. Data Packet Processing for Edge RBridge

The attached edge RBridge processes and forwards TRILL Data packets based on the endnode property rather than for encapsulation and forwarding the native frames the same way as the traditional RBridges. There are several situations for the edge RBridges as follows:

o If receiving an encapsulated unicast TRILL Data packet from a port with a Smart Endnode, with RB1's nickname as ingress, the edge RBridge RB1 forwards the frame to the specified egress nickname, as with any encapsulated frame. However, RB1 SHOULD filter the encapsulation frame based on the inner source MAC and Data Label as specified for the Smart Endnode. If the MAC (or Data Label) are not among the expected entries of the Smart Endnode, the frame would be dropped by the edge RBridge. If the edge RBridge does not perform this check, it makes it easier for a rogue end station to inject bogus TRILL Data packets into the TRILL campus.

o If receiving a unicast TRILL Data packet with RB1's nickname as egress from the TRILL campus, and the destination MAC address in the enclosed packet is a MAC address that has been listed by a "Smart Endnode", RB1 leaves the packet encapsulated to that Smart Endnode. The outer Ethernet destination MAC is the destination Smart Endnode's MAC address, the inner destination MAC address is either the Smart Endnode's MAC address or some other MAC address that the Smart Endnode advertised in its Smart Hello, and the outer Ethernet source MAC address is the RB1's port MAC address. The edge RBridge still decreases the Hop count value by 1, for there is one hop between the RB1 and Smart Endnode.

o If receiving a multi-destination TRILL Data packet from a port with a Smart Endnode, RBridge RB1 forwards the TRILL encapsulation to the TRILL campus based on the distribution tree indicated by the egress nickname. If the egress nickname does not correspond to a distribution tree, the packet is discarded. If there are any normal endnodes (i.e, non-Smart Endnodes) attached to the edge RBridge RB1, RB1 decapsulates the frame and sends the native frame to these ports possibly pruned based on multicast listeners, in addition to forwarding the multi-destination TRILL frame to the rest of the campus.

o If RB1 receives a native multi-destination data frame, which is sent by a non-Smart Endnode, from a port, including hybrid endnodes (Smart Endnodes and non-Smart Endnodes), RB1 will encapsulate it as multi-destination TRILL Data packet , and send the encapsulated multi-destination TRILL Data Packet out that same port to the Smart Endnodes attached to the port, and also send the encapsulated multi-destination TRILL Data Packet to the TRILL campus through other ports.

o If RB1 receives a multi-destination TRILL Data packet from a remote RBridge, and the exit port includes hybrid endnodes(Smart Endnodes and non-Smart Endnodes), it sends two copies of multicast frames out the port, one as native and the other as TRILL encapsulated frame. When Smart Endnode receives multi-destination TRILL Data packet, it learns the remote (MAC address, Data Label, Nickname) entry. A Smart Endnodes ignores native data frames. A normal (non-Smart) Endnode receives the native frame and learns the remote MAC address and ignores the TRILL data packet. This transit solution may bring some complexity for the edge RBridge and waste network bandwidth resource, so avoiding the hybrid endnodes scenario by attaching the Smart Endnodes and non-Smart Endnodes to different ports is RECOMMENDED.

6. Multi-homing Scenario

Multi-homing is a common scenario for the Smart Endnode. The Smart Endnode is on a link attached to the TRILL domain in two places: to edge RBridge RB1 and RB2. Take the figure below as example. The Smart Endnode SE1 is attached to the TRILL domain by RB1 and RB2 separately. Both RB1 and RB2 could announce their nicknames to SE1.

```
                     . .....................
                     .  +------+           .
                     .  | RB1  |           .
                     . /+------+           .
         +----------+ ./          +-----+ .    +----------+
         |SE1(Smart |/.           | RB3 |......| Smart    |
         | Endnode1)| .\          +-----+ .    | Endnode2 |
         +----------+ . \                 .    +----------+
                     .  +-----+           .
                     .  | RB2 |   TRILL   .
                     .  +-----+   Domain  .
                     .......................
```

Figure 4 Multi-homing Scenario

Smart Endnode SE1 can choose either RB1 or RB2’s nickname, when encapsulating and forwarding a TRILL data packet. If the active-active load balance is considered for the multi-homing scenario, the Smart Endnode SE1 could use both RB1 and RB2’s nickname to encapsulate and forward TRILL Data packet. SE1 uses RB1’s nickname when forwarding through RB1, and RB2’s nickname when forwarding through RB2. This will cause MAC flip-flopping(see [RFC7379]) of the endnode table entry in the remote RBridges (or Smart Endnodes). The solution for the MAC flip-flopping issue is to set a multi- homing bit in the RSV field of the TRILL data packet. When remote RBridge RB3 or Smart Endnodes receives a data packet with the multi-homed bit set, the endnode entries (SE1’s MAC address, label, RB1’s nickname) and (SE1’s MAC address, label, RB2’s nickname) will coexist as endnode entries in the remote RBridge. (An alternative solution would be to use the ESADI protocol to distribute multiple attachments of a MAC address of a multi-homing group, The ESADI is deployed among the edge RBridges (See section 5.3 of [RFC7357])).

7. Security Considerations

Smart-Hellos can be secured by using Authentication TLVs based on [RFC5310]. If they are not secured, then it is easier for a rogue end station that does not posses the required keying material to be falsely recognized as a valid Smart Endnode.

For general TRILL Security Considerations, see [RFC6325]. As stated there, since end stations are connected to edge RBridge ports by Ethernet, those ports MAY require end stations to authenticate themselves using [IEEE802.1X] and authenticate and encrypt traffic to/from the RBridge port with [IEEE802.1AE].

If they misbehave, Smart Endnodes can forge arbitrary ingress and egress nicknames in the TRILL Headers of the TRILL Data packets they construct. Decapsulating at egress RBridges or remote Smart Endnodes that believe such a forged ingress nickname would send future traffic destined for the inner source MAC address of the TRILL Data frame to the wrong edge RBridge if data plane learning is in use. Because of this, an RBridge port should not be configured to support Smart Endnodes unless the end stations on that link are trusted or can be adequately authenticated.

As with any end station, Smart Endnodes can forge the outer MAC addresses of packets they send (See Section 6 of [RFC6325].) Because they encapsulate TRILL Data packets, they can also forge inner MAC addresses. The encapsulation performed by Smart Endnodes also means they can send data in any Data Label which means they must be trusted in order to enforce a security policy based on Data Labels.

The TRILL-Hello is a type of TRILL ES-IS, and is defined in [RFC8171]. Receiving and processing TRILL-Hello for RBridges and Smart Endnodes would not bring more security and vulnerability issues than the TRILL ES-IS security defined in [RFC8171].

For added security against the compromise of data due to its mis-delivery for any reason, including the above, end-to-end encryption and authentication should be considered; that is, encryption and authentication from source end station to destination end station.

The mechanism described in this document requires Smart Endnodes to be aware of the MAC address(es) of the TRILL edge RBridge(s) to which they are attached and the egress RBridge nickname from which the destination of the packets is reachable. With that information, Smart Endnodes can learn a substantial amount about the topology of the TRILL domain. Therefore, there could be a potential security risk when the Smart Endnodes are not trusted or are compromised.

8. IANA Considerations

IANA is requested to allocate APPsub-TLV type numbers for the Smart-MAC and Smart-Parameters APPsub-TLVs from the range below 256 and update the "TRILL APPsub-TLV Types under IS-IS TLV 251 Application Identifier 1" registry as follows.

Protocol	Description	Reference
TBD1	Smart-Parameters	[this document]
TBD2	Smart-MAC	[this document]

Table 1

9. Acknowledgements

The contributions of the following persons are gratefully acknowledged: Mingui Zhang, Weiguo Hao, Linda Dunbar, Kesava Vijaya Krupakaran and Andrew Qu.

10. References

10.1. Informative References

[IEEE802.1AE]
"IEEE Standard for Local and metropolitan area networks--Media Access Control (MAC) Security.", 2006.

[IEEE802.1X]
"IEEE Standard for Local and metropolitan area networks--Port-Based Network Access Control", 2010.

[RFC7067] Dunbar, L., Eastlake 3rd, D., Perlman, R., and I. Gashinsky, "Directory Assistance Problem and High-Level Design Proposal", RFC 7067, DOI 10.17487/RFC7067, November 2013, <https://www.rfc-editor.org/info/rfc7067>.

[RFC7379] Li, Y., Hao, W., Perlman, R., Hudson, J., and H. Zhai, "Problem Statement and Goals for Active-Active Connection at the Transparent Interconnection of Lots of Links (TRILL) Edge", RFC 7379, DOI 10.17487/RFC7379, October 2014, <https://www.rfc-editor.org/info/rfc7379>.

10.2. Normative References

[IS-IS] ISO/IEC 10589:2002, Second Edition,, "Intermediate System to Intermediate System Intra-Domain Routing Exchange Protocol for use in Conjunction with the Protocol for Providing the Connectionless-mode Network Service (ISO 8473)", 2002.

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

[RFC5310] Bhatia, M., Manral, V., Li, T., Atkinson, R., White, R., and M. Fanto, "IS-IS Generic Cryptographic Authentication", RFC 5310, DOI 10.17487/RFC5310, February 2009, <https://www.rfc-editor.org/info/rfc5310>.

[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, <https://www.rfc-editor.org/info/rfc6325>.

[RFC7172] Eastlake 3rd, D., Zhang, M., Agarwal, P., Perlman, R., and D. Dutt, "Transparent Interconnection of Lots of Links (TRILL): Fine-Grained Labeling", RFC 7172, DOI 10.17487/RFC7172, May 2014, <https://www.rfc-editor.org/info/rfc7172>.

[RFC7176] Eastlake 3rd, D., Senevirathne, T., Ghanwani, A., Dutt, D., and A. Banerjee, "Transparent Interconnection of Lots of Links (TRILL) Use of IS-IS", RFC 7176, DOI 10.17487/RFC7176, May 2014, <https://www.rfc-editor.org/info/rfc7176>.

[RFC7357] Zhai, H., Hu, F., Perlman, R., Eastlake 3rd, D., and O. Stokes, "Transparent Interconnection of Lots of Links (TRILL): End Station Address Distribution Information (ESADI) Protocol", RFC 7357, DOI 10.17487/RFC7357, September 2014, <https://www.rfc-editor.org/info/rfc7357>.

[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, <https://www.rfc-editor.org/info/rfc7780>.

[RFC8171] Eastlake 3rd, D., Dunbar, L., Perlman, R., and Y. Li, "Transparent Interconnection of Lots of Links (TRILL): Edge Directory Assistance Mechanisms", RFC 8171, DOI 10.17487/RFC8171, June 2017, <https://www.rfc-editor.org/info/rfc8171>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

Authors’ Addresses

Radia Perlman
Dell EMC
176 South Street
Hopkinton, MA 01748
USA

Phone: +1-206-291-367
Email: radiaperlman@gmail.com

Fangwei Hu
ZTE Corporation
No.889 Bibo Rd
Shanghai 201203
China

Phone: +86 21 68896273
Email: hu.fangwei@zte.com.cn

Donald Eastlake
Huawei Technologies
155 Beaver Street
Milford, MA 01757
USA

Phone: +1-508-634-2066
Email: d3e3e3@gmail.com

Ting Liao
Huawei Technologies
Nanjing, Jiangsu 210012
China

Email: liaoting1@huawei.com

TRILL WG
INTERNET-DRAFT
Intended Status:Informational
Expires: October 20, 2017

R. Parameswaran,
Brocade Communications, Inc.
April 22, 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.

Status of This Memo

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.

```
    A--   --B
   / \ \/   /\
  /   \/\ _/_ \
 /___ _/\  /   \\
//      \/     \\
1        2       3
 \       |      /
  \      |     /
   \     |    /
    \    |   /
     \   |  /
      \  | /
       \ |/
         C
```

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.

```
                A
               /|\
              / | \
             /  |  \
            1   2   3
                /\
               /  \
              B    C
```

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.

```
                A
               / \
              /   \
             /     \
            2       3
                    /\
                   /  \
                  B    C
```

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

Copyright and IPR Provisions