INTERNET-DRAFT
Intended status: Proposed Standard
Expires: December 19, 2017

Donald Eastlake
Huawei

June 20, 2017

A Group Keying Protocol
<draft-eastlake-trill-group-keying-02.txt>

Abstract

This document specifies a general group keying protocol. It also provides use profiles for the application of this group keying protocol to multi-destination TRILL Extended RBridge Channel message security and TRILL over IP packet security.

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.

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

1. Introduction

This document specifies a general group keying protocol in Section 2. In addition, it provides, in Section 3, the use profile for the application of this group keying protocol to a case using DTLS (TRILL [RFC6325] [RFC7780] Extended RBridge Channel message security [RFC7178] [RFC7978]) and IPsec [TRILLoverIP}. It is anticipated that there will be other uses for this group keying protocol.

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] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document uses terminology and acronyms defined in [RFC6325] and [RFC7178]. Some of these are repeated below for convenience along with additional new terms and acronyms.

AES - Advanced Encryption Standard.

Data Label - VLAN or FGL.

DTLS - Datagram Transport Level Security [RFC6347].

FGL - Fine Grained Label [RFC7172].

GKd - A distinguished station in a group that is in charge of which group keying (Section 2) is in use.

GKs - Stations in a group other than GKd (Section 2).

HKDF - Hash based Key Derivation Function [RFC5869].

IS-IS - Intermediate System to Intermediate System [RFC7176].

keying material - The set of a Key ID, a secret key, and a cypher suite.

PDU - Protocol Data Unit.

QoS - Quality of Service.

RBridge - An alternative term for a TRILL switch.

SHA - Secure Hash Algorithm [RFC6234].

TRILL - Transparent Interconnection of Lots of Links or Tunneled Routing in the Link Layer.

TRILL switch - A device that implements the TRILL protocol [RFC6325] [RFC7780], sometimes referred to as an RBridge.

2. Group Keying Protocol

This section defines a general Group Keying Protocol that provides shared secret group keys. Any particular use of this protocol will require profiling giving further details and specifics for that use. The protocol is not suitable for discovery messages but is intended for use between members of a group that have already established pair-wise security.

2.1 Assumptions

The following are assumed:

- All pairs of stations in the group can engage in pairwise communication with unicast messages and each can groupcast a message to the other group members.
- At any particular time, there is a distinguished station GKd in the group that is in charge of keying for the groupcast data messages to be sent to the group. The group wide shared secret keys established by GKd are referred to herein as "dynamic" keys.
- Pairwise keying has been negotiated between GKd and each other station GKs1, GKs2, ... GKsN in the group. These keys are referred to in this protocol as "pairwise" keys.
- One or more keys, other than the dynamic or pairwise keys, each of which is already in place at all group member stations. These are referred to as "stable" keys.

When keying material is stored by a station, it is accompanied by a "use flag" indicating whether or not that keying material is usable for groupcast transmissions.

2.2 Group Keying Procedure Overview

GKd sends unicast keying messages to the other stations in the group and they respond as specified below and in further detail in the particular use profiles for this Group Keying Protocol. All such keying messages MUST be encrypted and authenticated using the pairwise keys as further specified in the use profile.

Typically, GKd sends a keying message to each GKs with keying material. After successful acknowledgement of receipt from each GKs, GKd sends a keying message to each GKs instructing it to use the dynamic key GKd has set. It would be common for GKd to set a new dynamic key at each GKs while an older dynamic key is in use so that GKd can more promptly roll over to the new key when appropriate.

To avoid an indefinite build up of keying material at a GKs, keys have a lifetime specified by GKd and GKd can send a message deleting a key. (GKd can also send a message indicating that a key is no longer to be used but leaving it set.) Should the space available at a GKs for keying material be exhausted, on receipt of a Set Key keying message for a new key ID GKs discards a dynamic key it has and originates a Delete Key message to the source of that dynamic key.

2.3 Transmission and Receipt of Group Data Messages

If a group has only two members, then pairwise security is used between them.

When a group has more than two members and a station in the group transmits a data message to the group, if the transmitter has one or more keys set by GKd that it has been instructed to use, it uses one of those keys and its associated cypher suite to groupcast the data message. If it has no such key, then it uses serial unicast to send the data message to each other member of the group, negotiating pairwise keys with them if it does not already have such pairwise keys. Thus it is a responsibility of GKd not to authorize the use of a groupcast key until it knows that all the GKs have that key.

When a station in the group receives data that has been groupcast to the group, if the receiver has the key referenced by the data message the receiver decrypts and verifies it. If verification fails or if the receiver does not have the required key, the receiver discards the data message. Thus whether GKs has been directed to "use" a key by GKd is relevant only to transmission, not reception.

2.4 Changes in Group Membership or GKd

When a new station joins the group, GKd should send that station the currently in-use group key and instruct it to use that key and send it other keys known to the group members and intended for future use.

If GKd detects that one or more stations that were members of the group are no longer members of the group, it SHOULD generate and distribute a new group key to the remaining group members, instruct them to use this new key, and delete from them any old keys known to the departed group member station(s) or at least instructing them to disuse such old keys that are marked for use; however, in the case of groups with large and/or highly dynamic membership, where a station might frequently leave and then rejoin, it may, as a practical matter, be necessary to rekey less frequently.

A new group member can become GKd due to the previous GKd leaving the group or a configuration change or the like. A GKs MUST NOT use keying material set by a station that it determines is not GKd. To avoid a gap in service, a station that is not GKd MAY set keying material at other stations in the group; however, such a non-GKd station cannot set the use flag for any such keying material. It is RECOMENDED that the second highest priority station to be GKd set such keying material at all other stations in the group. Should a station run out of room for keying material, it SHOULD discard keying material set by a station with lower priority to be GKd before discarding keying material set by a higher priority station and among keys set by GKd is SHOULD discard the lest recently used first.

2.5 Group Keying Messages

Keying messages start with a Version number. This document specifies Version zero.

Keying messages are structured as

- a Version number,
- a Response flag,
- a Key ID length,
- the Key ID of a stable key,
- a group keying use profile identifier,
- possible padding, and finally
- an AES key wrapped [RFC5649] [RFC3394] vector of additional fields wrapped using the stable key identified and using AES-256, as shown in Figure 2.1 below.

Keying messages are always sent unicast and encrypted and authenticated with the appropriate pairwise key, all as further specified for the particular use profile. It will typically be possible for GKd to calculate the keying message once, including the AES wrapping under a stable key, then send that message to various GKs using the different pairwise keys for each GKs.

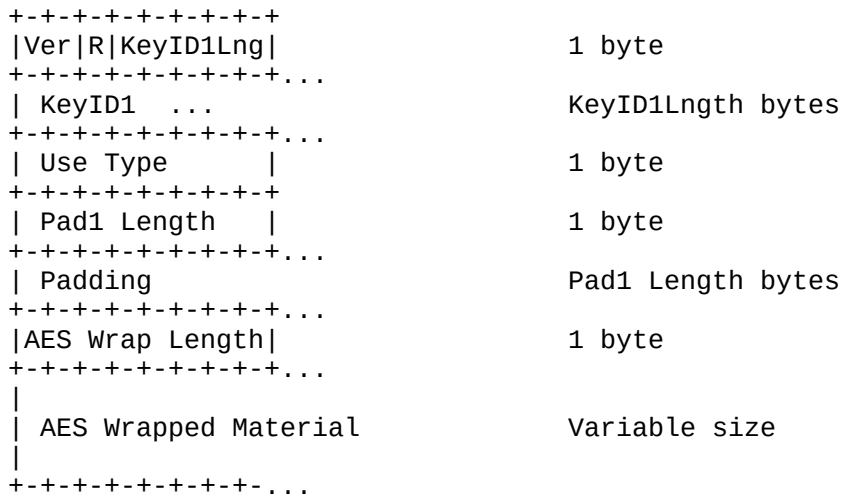


Figure 2.1. Keying Message Structure

The fields in Figure 2.1 are as follows:

Ver - Group Keying protocol version. This document specifies version zero.

R - Response flag. If set to one, indicates a response message. If set to zero, indicated a request or no-op message.

KeyID1Lngth, KeyID1 - KeyID1 identifies the stable AES-256 key wrapping key (also known as the Key Encrypting Key (KEK)) as further specified in the use profile. KeyID1Lngth is a 5-bit field that gives the length of KeyID1 in bytes as an unsigned integer.

Use Type - Specifies the particular group security use profile such as RBridge Extension (Section 3) or IP link [TRILLoverIP].

Pad1 Length, Pad1 - Padding to obscure the non-padded message size. Pad1 Length may be from 0 to 255 and gives the length of the padding as an unsigned integer. Each byte of padding MUST be equal to Pad1 Length. For example, 3 bytes of padding with length is 0x03030303.

AES Wrap Length - An unsigned byte that gives the length of the AES Wrapped Material in units of 8 bytes. The length of AES key wrapped material is, as specified in [RFC5649], always a multiple of 8 bytes (64 bits) and not less than 16 bytes. Thus an AES Wrap Length of 0 or 1 is invalid.

AES Wrapped Material - The output of the AES Key Wrapping operation on the message vector of fields using the specified stable key.

The vector of fields contained within the AES-256 key wrapping is specified for the various keying messages in subsections below. The contents of this wrapped vector are protected by the AES wrapping as well as being authenticated and super-encrypted by the pairwise keyed security used for sending the overall keying message. The stable key used for AES wrapping MUST be different from the outer message pairwise key.

Each group keying message contains, in the AES wrapped vector of fields, a message type and a message ID set by the sender of a request. These fields are returned in the corresponding response to assist in the matching of response to requests, except that there is no response to the No-Op message.

If no response is received to a request (other than a No-Op message) for an amount of time configurable in milliseconds from 1 to (2**15 - 1), the request is re-transmitted with the same message ID. These retries can occur up to a configurable number of times from 1 to 8. Unless otherwise provided in the particular use profile, the default response delay threshold is 200 milliseconds and the default maximum number of retries is 3.

Keying messages are sent with a priority/QoS configurable on a per device per use type basis. The default priority/QoS is specified in the use profile.

Since the minimum length of the AES Wrapped Material is 16 bytes [RFC5649], the minimum valid size of a keying message is 20 bytes, even if KeyID1 Length and Pad1 Length are zero. All multi-byte fields are in network order, that is, with the most significant byte first.

2.6 Set Key Message

The structure of the wrapped vector of fields for the Set Key keying message is as show in Figure 2.2. A recipient automatically determines the overall length provided for this vector of fields inside the AES wrapping as a byproduct of the process of AES unwrapping [RFC5649].

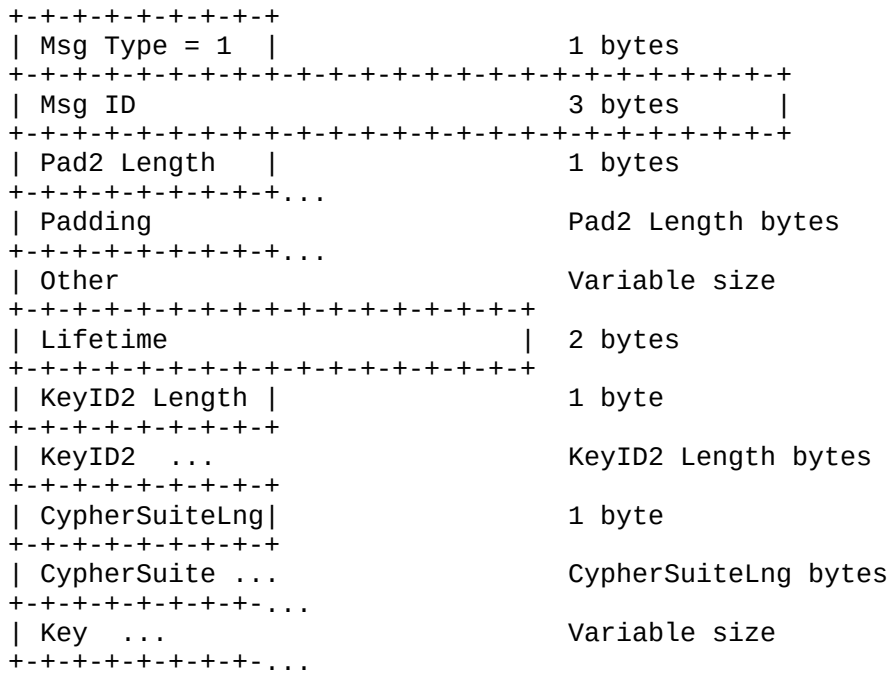


Figure 2.2. Set Key Message Inner Structure

The fields are as follows:

Msg Type = 1 for Set Key message

Msg ID - A 3 byte quantity to be included in the corresponding response message to assist in matching requests and responses. Msg ID zero has a special meaning in responses and MUST NOT be used in a Set Key message or any other group keying request message.

Pad2 Length, Pad2 - Padding to obscure the size of the unapdded AES wrapped data. Pad2 Length may be from 0 to 255 and gives the length of the padding as an unsigned integer. Each byte of padding MUST be equal to Pad1 Length. For example, 2 bytes of padding with length byte is 0x020202.

Other - Additional information if specified in the use profile. If Other information in this message is not mentioned in the use profile, there is none and this portion of the wrapped information is null. If a use profile specifies Other information it must be possible to determine its length so that following fields can be properly parsed and so that the size of the Key field can be deduced; for example, it could begin with a length byte.

Lifetime - A 2-byte unsigned integer. After that number of seconds plus one second, the key and associated information being set MUST be discarded. Unless otherwise specified for a particular use profile of this group keying protocol, the default Lifetime is 15,000 seconds or a little over four hours.

KeyID2 Length, KeyID2 - KeyID2 identifies the group key and associated information being set as further specified in the use profile. KeyID2 Length is an unsigned byte that gives the length of KeyID2 in bytes.

CypherSuiteLng, CypherSuite - CypherSuite identifies the cypher suite associated with the key being set as further specified in the use profile. CypherSuite Length is an unsigned byte the gives the length of CypherSuite in bytes.

Key - This is the actually group shared secret keying material being set. Its length is deduced from the overall length of the vector of fields (found by the AES unwrap operation) and the length of the preceding fields.

If GKs already has a dynamic key set under KeyID2, the key’s value and associated cypher suite are compared with those in the Set Key messages. If they are the same, the only receiver action is to update the Lifetime information associated with KeyID2 and send a Response message. If they are different, the lifetime, cypher suite, and key (and possibly Other material) are replaced, the use flag is cleared, and a Response message sent.

2.7 Use, Delete, Disuse, or Deleted Key Messages

The structure of the wrapped material for the Use Key, Delete Key, and Disuse Key keying messages are the same as each other except for the message type. This structure is shown in Figure 2.3

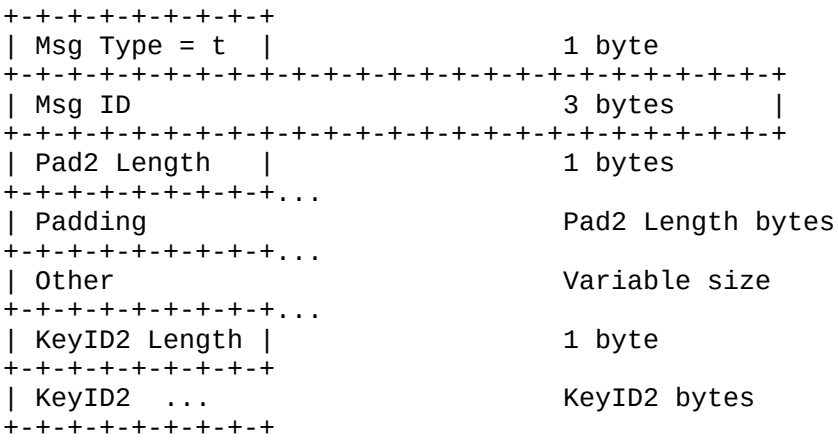


Figure 2.3. Use, Delete, Disuse, or Deleted Key Message

The Msg Type field specifies the particular message as follows:

Msg Type	Message
2	Use Key
3	Delete Key
4	Disuse Key
5	Deleted Key

The remaining fields are as specified in Section 2.4. KeyID2 indicates the key to be used, deleted, for which use should cease, or which has been deleted, depending on the message type.

It is RECOMMENDED that these messages be padded so as to be the same length as a typical Set Key message.

The Delete Key is sent by a station believing itself to be GKd instructing some GKs to delete a key. When a GKs spontaneously deletes a key, it sends a Deleted Key message to the station from which it received the key. The message types for Delete Key and Deleted Key are different to minimize confusion in corner cases such as the GKd changing while messages are in flight. The Msg ID used in a Deleted Key message is created by the sending GKs from a space of Msg IDs associated with that GKs which is independent of the Msg IDs used in requests originated by GKd.

2.8 Response Message

The structure of the wrapped material for the Response group keying message is as show below in Figure 2.4. A response message is

indicated by the R bit in the first byte of the message outside the key wrapping.

A response MUST NOT be sent due to the receipt of a response. The R bit is outside of the key wrapping so that this rule can be enforced even in cases of difficulty in unwrapping.

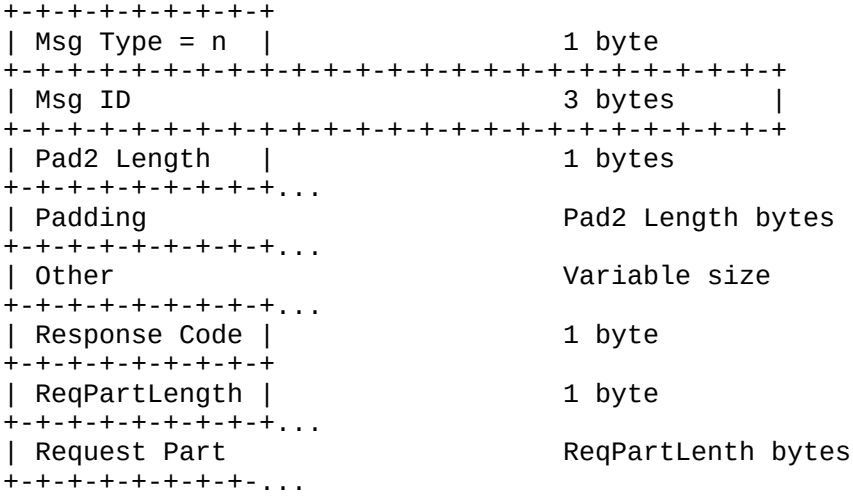


Figure 2.4. Response Message Inner Structure

Except as specified below, the fields are as specified for the Key Set message.

Msg Type, Msg ID - The content of these field is copied from the message in reply to which this Response message is sent unless there is an error that stops the replying station from determining them; in that case the special value zero is used for the Msg Type and Msg ID. Errors where the Msg Type and ID could not be determined are indicated by a Response Code with its high order bit set to one, that is, the 0b1xxxxxxx bit set.

Response Code - An unsigned byte giving the response as enumerated in Table 2.2 in Section 2.8.1. Any Response Code other than a success indicates that the receiver took no action on the request other than sending an error Response message.

ReqPartLength, Request Part: It is usually usefully to include some or all of the request message in error responses.

- If the Response Code high order two bits are zero, the request succeeded and ReqPartLength MUST be set to zero so Request Part will be null.

- If the Response Code high order two bits are zero one (0b01xxxxxx), then there was an error in the part of the request inside the AES key wrapping but the unwrap process was successful. ReqPartLength is the length of the request message material included in the Request Part field. The included request material is from the unwrapped vector of fields started with the Msg Type byte.
- If the Response Code high order bit is one (the 0b1xxxxxxx is set), then there was an error parsing the material outside the AES key wrap or an error in the AES unwrapping process. ReqPartLength is the length of the request message part included in the Request Part field. The included part of the request starts with the first byte of the message (the byte containing the version, response flag, and KeyID1 Length).

2.8.1 Response Codes

The high order two bits of the Response Code have meaning as shown in Table 2.1.

Top 2 Bits	Category
0b00	Success
0b01	AES wrap contents
0b10/11	Outside of AES wrap contents

Response Decimal	Response Hex	Meaning
0	0x00	Success
1	0x01	Success and the key at an existing key ID was changed
2-47	0x02-0x2F	Unassigned
48-63	0x30-0x3F	Reserved for special success codes defined in use profiles
64	0x40	Malformed inner fields (see Note 2 below)
65	0x41	Unknown or zero Msg Type in a request
66	0x42	Zero Msg ID in a request
68	0x43	Invalid length KeyID2
69	0x44	Unknown KeyID2
70	0x45	Invalid length CypherSuite
71	0x46	Unknown CyperSuite
72	0x47	Bad Key (see Note 3 below)
73-111	0x49-0x6F	Unassigned
112-127	0x70-0x7F	Reserved for error codes defined in use profiles and related to the AES wrapped

		contents
128	0x80	Malformed message (see Note 1 below)
129	0x81	Invalid length KeyID1
130	0x82	Unknown KeyID1
131	0x83	Unknown Use Type
131	0x84	AES unwrap fails test 1, see Section 3 [RFC5649]
132	0x85	AES unwrap fails test 2, see Section 3 [RFC5649]
133	0x86	AES unwrap fails test 3, see Section 3 [RFC5649]
134-175	0x86-0x7F	Unassigned
176-191	0xB0-0xBF	Reserved for error codes defined in use profiles and related to parts of message outside the AES wrap contents
192	0xC0	No keys set
193	0xC1	Referenced key unknown
194	0xC2	Referenced key known but use flag not set
195-255	0xC3-0xFF	Reserved

Response Code Notes:

Note 1 Message is too short or too long, AES wrapped material is too short, Padding bytes are not the required value, or similar fundamental message format problems.

Note 2 The AES wrapped inner vector of fields is too short or too long, Padding bytes are not the required value, or similar fundamental vector of fields format problems.

Note 3 Key is not a valid length for CypherSuite or other internal checks on key (for example, parity bits in a 64 bit DES key (not that you should be using DES)) fail.

2.8 No-Op Message

The No-Op message is a dummy message intended for use in disguising metadata deducable from keying message transmissions. It requires no response although a recipient can always decided to send a No-Op message to a station from which it has received such a message. The vector of fields inside the AES key wrap is as follows:

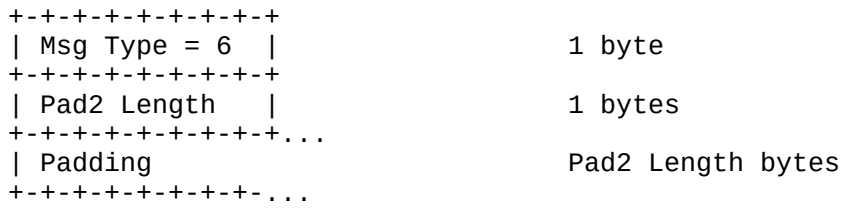


Figure 2.5. No-Op Message Inner Structure

The Msg Type is set to 6 to indicate a No-Op message.

Pad2 Length and Padding are as specified in Section 2.6. It is RECOMMENDED that Pad2 Length in a No-Op message be such as to make its length the same as the length of a typical Set Key message.

2.9 General Security Considerations

This section gives some general security considerations of this group keying protocol as distinguished from security considerations of a particular use profile.

The method by which the stations in the group discover each other is specified in the group keying use profile. GKd controls group access and generally learns whatever it needs to know about GKs during the pairwise authentication and pairwise keying process.

The group keying provided by this protocol is shared secret keying. This means that data messages can only be authenticated as coming from some group member but not as coming from a specific group member. If this level of authentication is insufficient, GKd can simply not set keys or not set them as usable. This will force all stations in the group that are configured to use security for multi-destination transmissions to the group to serial unicast data to the other group members using pairwise keying.

The content value of padding fields in the Group Keying protocol is fixed so that it cannot be used as a covert channel. The length of padding could still be so used.

3. DTLS: Extended RBridge Channel Group Keyed Security

This section specifies a profile of the group keying protocol defined in Section 2. This profile provides shared secret keying to secure multi-destination Extended RBridge Channel messages [RFC7978]. The keys put in place by the group keying protocol are available for use as DTLS pre-shared keys with the DTLS and Composite Security of multi-destination Extended RBridge Channel messages as specified in Section 3.2.

For this group keying use profile, a group is identified by TRILL Data Label (VLAN or FGL [RFC7172]) and consists of the data reachable [RFC7780] RBridges with interest in that Data Label. GKd is the RBridge in the group that, of those group members supporting the Group Keying Protocol, is the highest priority to be a TRILL distribution tree root. If not all members of the group support the Group Keying Protocol, then there are two cases for multi-destination Channel Tunnel RBridge Channel messages:

(1) If the sender and at least two other group members support the Group Keying Protocol, it SHOULD, for efficiency, send a secured multi-destination RBridge Channel message to cover the group and serially unicast to the group members not supporting the Group Keying Protocol.
(2) In other cases the sender serially transmits the data to the group members using pairwise security.

3.1 Transmission of Group Keying Messages

Keying messages themselves are sent as unicast Extended RBridge Channel messages carrying a Group Keying protocol (see Section 5.2) RBridge Channel message. They MUST use DTLS Pairwise or Composite (STypes 2 or 3) security.

The Group Keying profile for this Group Keying Use Type is as follows:

Priority of Group Keying messages for this SHOULD be 6 unless the network manager chooses to use a lower priority after determining that such lower priority group keying messages will yield acceptable performance. Priority 7 SHOULD NOT be used as it may cause interference with the establishment and maintenance of adjacency.

Use Type = 1

KeyID1 Length = 2, KeyID1 is an [RFC5310] key ID.

CypherSuiteLng = 2, CypherSuite is the cypher suite used in

groupcast extended RBridge Channel data messages for the corresponding KeyID2. This a DTLS [RFC6347] cypher suite.

KeyID2 Length = 1, KeyID2 is the index under which a group key is set. Group keys are, in effect, indexed by this KeyID2 and the nickname of the GKd as used in the Ingress Nickname field of the TRILL Header of Group Keying messages.

3.2 Transmission of Protected Multi-destination Data

Protected Extended RBridge Channel [RFC7978] messages are multicast (M bit set to one in the TRILL Header) and set the SType field to a new value for "Group Secured" (See Section 5.3). The data is formatted as one byte of Key ID followed by data formatted as TLS 1.2 [RFC5246] application_data using the cyphersuite and keying material stored under the Key ID.

4. TRILL Over IP Group Keyed Security

This section specifies a profile of the group keying protocol defined in Section 2. This profile provides shared secret keying to secure TRILL over IP messages [TRILLoverIP]. The keys put in place by the group keying protocol are available for use as IPSEC keys.

For this group keying use profile, a group is identified by an IP multicast address and consists of the adjacent [RFC7177] RBridges reachable with that multicast address. GKd is the RBridge in the group that, of those group members supporting the Group Keying Protocol, has the highest priority to be a TRILL distribution tree root. If not all members of the group support the Group Keying Protocol, then there are two cases for multi-destination TRILL over IP messages:

(1) If the sender and at least two other group members support the Group Keying Protocol, it SHOULD, for efficiency, send a secured IPSEC message to cover the group and serially unicast to the group members not supporting the Group Keying Protocol.
(2) In other cases the sender serially transmits the data to the group members using pairwise security.

4.1 Transmission of Group Keying Messages

tbd

Use Type = 2

tbd

4.2 Transmission of Protected Multi-destination Data

tbd

5. IANA Considerations

This section gives IANA Considerations.

5.1 Group Keying Protocol

IANA is requested to perform the following actions:

1. Establish a protocol parameters web page for "Group Keying Protocol Parameters" with the initial registries on that page as specified below in this section.

2. Establish a "Message Type" registry on the Group Keying Protocol Parameters page as follows:

   Registration Procedure: IETF Review

   Reference: [this document]

Type	Description	Reference
0	Reserved	[This document]
1	Set Key	[This document]
2	Use Key	[This document]
3	Delete Key	[This document]
4	Disuse Key	[This document]
5	Deleted Key	[This document]
6	No-Op	[This document]
7-250	Unassigned	
251-254	Reserved for Private Use	[This document]
255	Reserved	[This document]

3. Establish a "Group Keying Use Profile" registry on the Group Keying Protocol Parameters page as follows:

   Registration Procedure: IETF Review

   Reference: [This document]

Profile	Description	Reference
0	Reserved	[This document]
1	Extended RBridge Channel	[This document]
2	TRILL over IP	[This document]
3-250	Unassigned	
251-254	Reserved for Private Use	[This document]
255	Reserved	[This document]

4. Establish a "Response Code" registry on the Group Keying Protocol Parameters page as show below taking entries from the Response Code table in Section 2.8.1 above. In the table of values, the Reference column should be "[This document]" except where the Meaning is "Unassigned" or "Reserved".

   Registration Procedure: IETF Review

   Reference: [This document]

   Note: The top two bits of the Response Code indicate a category as specified in Section 2.8.1 of [this document].

Response Decimal	Response Hex	Meaning	Reference
0	0x00	Success	[this document]
...	...	...	
255	0xFF	Reserved	

5.2 Group Keying RBridge Channel Protocol Numbers

IANA is requested to assign TBD1 as the TRILL RBridge Channel protocol number, from the range assigned by Standards Action, for use when the "Group Keying" protocol is transmitted over Extended RBridge Channel messages.

The added RBridge Channel protocols registry entry on the TRILL Parameters web page is as follows:

Protocol	Description	Reference
TBD1	Group Keying	Section 2 of [this document]

5.3 Group Secured Extended RBridge Channel SType

IANA is requested to assign TBD2 as the Group Secured SType in the "Extended RBridge Channel Security Types Subregistry" on the TRILL Parameters web page as follows:

SType	Description	Reference
TBD2	Group Secured	Section 3.2 of [this document]

6. Security Considerations

TBD

See [RFC7978] for Extended RBridge Channel security.

See [RFC7457] in connection with TLS and DTLS security.

Normative References

[RFC2119] - BBradner, 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>.

[RFC3394] - Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, DOI 10.17487/RFC3394, September 2002, <http://www.rfc-editor.org/info/rfc3394>.

[RFC5246] - Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[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, <http://www.rfc-editor.org/info/rfc5310>.

[RFC5649] - Housley, R. and M. Dworkin, "Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm", RFC 5649, DOI 10.17487/RFC5649, September 2009, <http://www.rfc-editor.org/info/rfc5649>.

[RFC5869] - Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, May 2010, <http://www.rfc-editor.org/info/rfc5869>.

[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>.

[RFC6347] - Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[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, <http://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, May 2014, <http://www.rfc-editor.org/info/rfc7176>.

[RFC7177] - Eastlake 3rd, D., Perlman, R., Ghanwani, A., Yang, H., and V. Manral, "Transparent Interconnection of Lots of Links

(TRILL): Adjacency", RFC 7177, DOI 10.17487/RFC7177, May 2014, <http://www.rfc-editor.org/info/rfc7177>.

[RFC7178] - Eastlake 3rd, D., Manral, V., Li, Y., Aldrin, S., and D. Ward, "Transparent Interconnection of Lots of Links (TRILL): RBridge Channel Support", RFC 7178, DOI 10.17487/RFC7178, May 2014, <http://www.rfc-editor.org/info/rfc7178>.

[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>.

[RFC7978] - Eastlake 3rd, D., Umair, M., and Y. Li, "Transparent Interconnection of Lots of Links (TRILL): RBridge Channel Header Extension", RFC 7978, DOI 10.17487/RFC7978, September 2016, <http://www.rfc-editor.org/info/rfc7978>.

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

[TRILLoverIP] - M. Cullen, D. Eastlake, M. Zhang, D. Zhang, "Transparent Interconnection of Lots of Links (TRILL) over IP", draft-ietf-trill-over-ip, work in progress.

Informative References

[RFC6234] - Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <http://www.rfc-editor.org/info/rfc6234>.

[RFC7457] - Sheffer, Y., Holz, R., and P. Saint-Andre, "Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS)", RFC 7457, February 2015, <http://www.rfc-editor.org/info/rfc7457>.

Acknowledgements

The contributions of the following are hereby gratefully acknowledged:

TBD

The document was prepared in raw nroff. All macros used were defined within the source file.

Authors' Addresses

Donald E. Eastlake, 3rd
Huawei Technologies
155 Beaver Street
Milford, MA 01757 USA

Phone: +1-508-333-2270
EMail: d3e3e3@gmail.com

Copyright, Disclaimer, and Additional IPR Provisions

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."

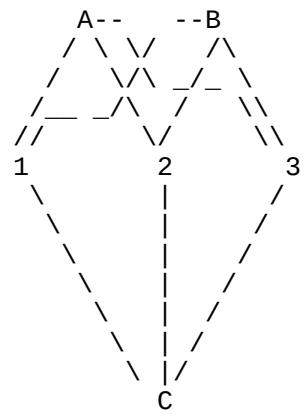
There is an additional correction posted to this in [RFC7780]:

[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.



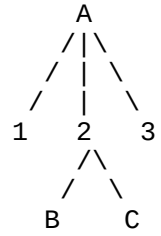
Assume that in the above topology, when ordered by 7-octet ISIS-id,

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.

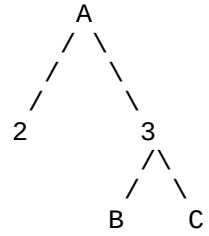


However, due to TRILL's parent selection algorithm, the sub-tree rooted at Node 2 will be impacted even if Node 1 or Node 3 go down.

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