Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2015

N. Akiya
C. Pignataro
D. Ward
Cisco Systems

October 21, 2014

Seamless Bidirectional Forwarding Detection (S-BFD) Alert Discriminator
draft-akiya-bfd-seamless-alert-discrim-03

Abstract

This document defines the Alert Discriminator which operates on the Seamless Bidirectional Forwarding Detection (S-BFD), and Alert Discriminator Diagnostic Codes which operates on the Alert Discriminator.

Requirements Language

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 RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on April 24, 2015.

Copyright Notice

Table of Contents

1. Introduction

[I-D.ietf-bfd-seamless-base] defines the Seamless Bidirectional Forwarding Detection (S-BFD): a simplified mechanism which uses Bidirectional Forwarding Detection (BFD) with large portions of negotiation aspects eliminated.

This document defines the Alert Discriminator which operates on the S-BFD, and the Alert Discriminator Diagnostic Codes which operates on the Alert Discriminator, for extended S-BFD use cases described in Section 2.

2. Extended S-BFD Use Cases

This section describes extended S-BFD use cases.

2.1. Target S-BFD Discriminator Discovery

IS-IS ([I-D.ietf-isis-sbfd-discriminator]) and OSPF ([I-D.ietf-ospf-sbfd-discriminator]) protocols have been extended to advertise S-BFD discriminator values. These extensions will suffice for number of scenarios where S-BFD is used to verify the network reachability to other network devices. Other protocols may be extended to support S-BFD in further scenarios.

There are, however, some scenarios where it is desirable to have a mechanism within the S-BFD protocol to discover the target S-BFD discriminator value.

- In some scenarios, direct protocol communications are intentionally kept minimal for reasons such as administrative policy. One such example is the usage of S-BFD across Autonomous System (AS) boundaries (i.e. inter-AS).

- In some scenarios, there is no control plane which can easily advertise S-BFD discriminators. MPLS-TP and static routes are such examples.

- In some scenarios, defining and standardizing protocol extensions to advertise S-BFD discriminator values may be more work than the value it brings.

To accommodate the two scenarios described, it is desirable to have a mechanism within the S-BFD protocol to discover the target S-BFD discriminator value.

2.2. S-BFD Path Tracing

When a multihop S-BFD session, IP based or MPLS based, determines a loss of reachability to the target entity, the responsibility of identifying the problematic point in the paths is often left to operators. ICMP echo request/reply (IP Ping/Trace) [RFC0792] and MPLS echo request/reply (LSP Ping/Trace) [RFC4379] allow for tracing of hops to a specific target, and these are often used by operators, manually or automatically, to attempt to isolate faults. However, when it comes to identifying the problematic point that caused the S-BFD session to declare the failure, there are couple of issues.

- Usage of non-S-BFD packets can result in them being load balanced differently along the paths, causing those packets to traverse different paths than S-BFD packets did.

- Usage of non-S-BFD packets may not identify the problematic points which only affect specific flows (which affects S-BFD packets).

- In order to isolate short lived transient issues, it is desirable to immediately perform the task of fault isolation. IP/MPLS Ping/Trace implementations often require more processing overhead than S-BFD. Usage of heavier tool to attempt to isolate fault can result in missing more instances of identifying short lived transient issues.

Although the task of "fault isolation" does not belong in the BFD/S-BFD protocols, if the task of "fault isolation" can be done with simple extensions within the S-BFD protocol, the result does provide additional benefit to operators.

3. Alert Discriminator

This document reserves the value zero of the S-BFD discriminator pool as the Alert Discriminator. A reflector BFD session is to monitor incoming S-BFD packets with value zero in the "Your Discriminator" field. The reflector BFD session is to process the S-BFD packets according to the value specified in the received "Diagnostic" field. Procedures specific to each "Diagnostic" code are described in Section 4.

4. Alert Discriminator Diagnostic Codes

This section defines the Alert Discriminator Diagnostic Codes, and procedures for each defined code point. The Alert Discriminator Diagnostic Codes MUST operate on the Alert Discriminator. Specifically:

- In the direction from an SBFDInitiator to an SBFDReflector, the Alert Discriminator Diagnostic Codes MUST only be used with "Your Discriminator" field set to the Alert Discriminator.

- In the direction from an SBFDReflector to an SBFDInitiator, the Alert Discriminator Diagnostic Code MUST only be used in a reply S-BFD packet if received S-BFD packet contained "Your Discriminator" field set to the Alert Discriminator.

4.1. Diagnostic Code: Target S-BFD Discriminator Discovery

The Alert Discriminator Diagnostic Code 29 is defined for the purpose of discovering the target S-BFD discriminator.

Value	Alert Discriminator Diagnostic Code Name
29	Target S-BFD Discriminator Discovery

When a reflector BFD session receives an S-BFD packet containing the Alert Discriminator and the Alert Discriminator Diagnostic Code of 29, then the reflector BFD session SHOULD send a reply S-BFD packet. The format and the contents of the generated reply S-BFD packet MUST follow the definition in the S-BFD protocol documents, except for following fields:

- "My Discriminator" field MUST be set to one of local S-BFD discriminators.
- "Diagnostic" field MUST be set to value 29.

4.2. Diagnostic Code: S-BFD Path Tracing

The Alert Discriminator Diagnostic Code 30 is defined for the purpose of S-BFD path tracing.

Value	Alert Discriminator Diagnostic Code Name
30	S-BFD Path Trace

When a reflector BFD session receives an S-BFD packet containing the Alert Discriminator and the Alert Discriminator Diagnostic Code of 30, then the reflector BFD session SHOULD send a reply S-BFD packet. The format and the contents of the generated reply S-BFD packet MUST follow the definition in the S-BFD protocol documents, except for following fields:

- "My Discriminator" field MUST be set to zero.
- "Diagnostic" field MUST be set to value 30.

4.3. Diagnostic Code: Not Supported

The Alert Discriminator Diagnostic Code 31 is defined for a reflector BFD session to communicate, in reply S-BFD packet, that specified Alert Discriminator Diagnostic Code in received S-BFD packet is not understood or is not supported.

Value	Alert Discriminator Diagnostic Code Name
31	Not Supported

When a reflector BFD session receives an S-BFD packet containing the Alert Discriminator and an Alert Discriminator Diagnostic Code which is not understood or supported by the reflector BFD session, then the reflector BFD session SHOULD send a reply S-BFD packet. The format and the contents of the generated reply S-BFD packet MUST follow the

definition in the S-BFD protocol documents, except for following fields:

- "My Discriminator" field MUST be set to zero.

- "Diagnostic" field MUST be set to value 31.

Note that in the direction from an SBFDInitiator to an SBFDReflector, the Alert Discriminator Diagnostic Code 31 MUST NOT be used. If a reflector BFD session receives an S-BFD packet with the Alert Discriminator and the Alert Discriminator Diagnostic Code 31, then the reflector BFD session MUST drop the packet.

5. Security Considerations

Conceptually the Alert Discriminator is similar to an IP Router Alert Option or an MPLS Router Alert Label. The Alert Discriminator introduces a way which remote network devices can instruct a reflector BFD sessions to perform specific tasks corresponding to specified Alert Discriminator Diagnostic Codes, and without remote network devices knowing a valid S-BFD discriminator on the target device. Hence, it is very critical that reflector BFD session services the Alert Discriminator only from trusted sources and for allowed Alert Diagnostic Codes for those sources. Therefore, this document RECOMMENDS following security procedures to be implemented:

- S-BFD packets with Alert Discriminator is accepted only from trusted sources. An implementation SHOULD provide a mechanism for operators to specify an access-list to describe the trusted sources.

- An implementation SHOULD provide a mechanism for operators to specify the Alert Discriminator Diagnostic Codes which are supported on the device. If required, such configuration should be set per a trusted source.

Additionally, it is RECOMMENDED that implementations supporting the Alert Discriminator considers the security considerations described in [I-D.ietf-bfd-seamless-base], [I-D.ietf-bfd-seamless-ip] and [I-D.akiya-bfd-seamless-sr] documents.

6. IANA Considerations

This document requests IANA to create a new registry within [IANA-BFD] protocol to maintain "Alert Discriminator Diagnostic Codes" field. Initial values are described in immediate sub-section to follow.

6.1. Alert Discriminator Diagnostic Codes Registry

The IANA is requested to create and maintain a registry entitled "Alert Discriminator Diagnostic Codes" with the following registration procedures:

Registry Name: Alert Discriminator Diagnostic Codes

Value	Alert Discriminator Diagnostic Code Name	Reference
0-7	Experimental	This document
8-28	Reserved	This document
29	Target S-BFD Discriminator Discovery	This document
30	S-BFD Path Trace	This document
31	Not Supported	This document

Assignments of Alert Discriminator Diagnostic Codes are via Standards Action [RFC5226].

7. Acknowledgements

Authors would like to thank Srihari Raghavan and Girija Raghavendra Rao for reviewing and providing comments on this document.

8. Contributing Authors

Nagendra Kumar
Cisco Systems
Email: naikumar@cisco.com

Mallik Mudigonda
Cisco Systems
Email: mmudigon@cisco.com

Aswatnarayan Raghuram
AT&T
Email: ar2521@att.com

Glenward D. Hayden
AT&T
Email: gh1691@att.com

9. References

9.1. Normative References

[I-D.akiya-bfd-seamless-sr]
Akiya, N., Pignataro, C., and N. Kumar, "Seamless Bidirectional Forwarding Detection (S-BFD) for Segment Routing", draft-akiya-bfd-seamless-sr-03 (work in progress), August 2014.

[I-D.ietf-bfd-seamless-base]
Akiya, N., Pignataro, C., Ward, D., Bhatia, M., and J. Networks, "Seamless Bidirectional Forwarding Detection (S-BFD)", draft-ietf-bfd-seamless-base-03 (work in progress), August 2014.

[I-D.ietf-bfd-seamless-ip]
Akiya, N., Pignataro, C., and D. Ward, "Seamless Bidirectional Forwarding Detection (S-BFD) for IPv4, IPv6 and MPLS", draft-ietf-bfd-seamless-ip-00 (work in progress), September 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[I-D.ietf-isis-sbfd-discriminator]
Ginsberg, L., Akiya, N., and M. Chen, "Advertising S-BFD Discriminators in IS-IS", draft-ietf-isis-sbfd-discriminator-01 (work in progress), October 2014.

[I-D.ietf-ospf-sbfd-discriminator]
Bhatia, M., Pignataro, C., Aldrin, S., and T. Ranganath, "OSPF extensions to advertise S-BFD Target Discriminator", draft-ietf-ospf-sbfd-discriminator-00 (work in progress), September 2014.

[IANA-BFD]
IANA, "Bidirectional Forwarding Detection (BFD) Parameters", <http://www.iana.org/assignments/bfd-parameters/bfd-parameters.xhtml>.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", RFC 4379, February 2006.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

Authors' Addresses

Nobo Akiya
Cisco Systems

Email: nobo@cisco.com

Carlos Pignataro
Cisco Systems

Email: cpignata@cisco.com

Dave Ward
Cisco Systems

Email: wardd@cisco.com

Routing Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 23, 2015

A. Mishra
M. Jethanandani
A. Saxena
Ciena Corporation
S. Pallagatti
Juniper Networks
M. Chen
Huawei

October 20, 2014

BFD Stability
draft-ashesh-bfd-stability-01.txt

Abstract

This document describes extensions to the Bidirectional Forwarding Detection (BFD) protocol to measure BFD stability. Specifically, it describes a mechanism for detection of BFD frame loss.

Requirements Language

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 <xref target="RFC2119">RFC 2119</xref>.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on April 23, 2015.

Table of Contents

1. Introduction

The Bidirectional Forwarding Detection (BFD) protocol operates by transmitting and receiving control frames, generally at high frequency, over the datapath being monitored. In order to prevent significant data loss due to a datapath failure, the tolerance for lost or delayed frames (the Detection Time as described in RFC 5880) is set to the smallest feasible value.

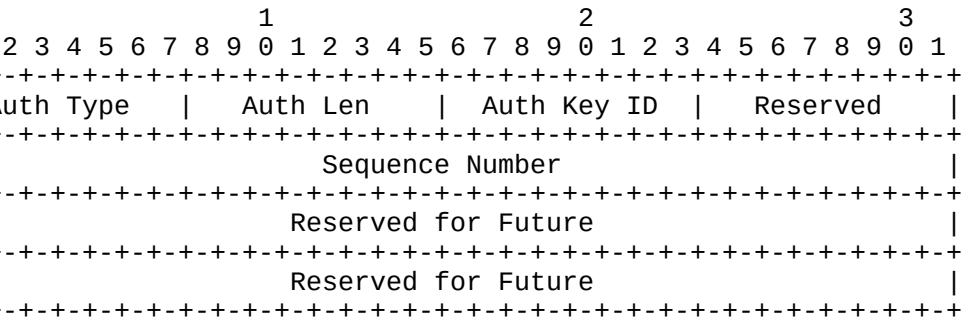
This document proposes a mechanism to detect lost frames in a BFD session in addition to the datapath fault detection mechanisms of BFD. Such a mechanism presents significant value with the ability to measure the stability of BFD sessions and provides data to the operators.

2. BFD Null-Authentication TLV

The functionality proposed for BFD stability measurement is achieved by appending the Null-Authentication TLV to the BFD control frame.

The Null-Authentication TLV (called 0-Auth in this document) extends the existing BFD Authentication TLV structure by adding a new Auth-

Type of <IANA Assigned>. This TLV carries the Sequence Number for frame loss measurement.



where:

Auth Type: The Authentication Type, which in this case is <IANA assigned> (Null Authentication).

Auth Len: The length of the Authentication Section, in bytes. Set to 12.

Auth Key ID: The Authentication Key ID in use for this packet. This MUST be set to zero on transmit, and ignored on receipt.

Reserved: This byte MUST be set to zero on transmit, and ignored on receipt.

Reserved for Future: For future extensions. MUST be set to 0 if not used.

Sequence Number: This indicates the sequence number for this packet and MUST be present in every 0-Auth TLV. This value is incremented by 1 for every frame transmitted while the session state is UP. A value of 0 indicates a request by sender to reset the sequence number correlation logic at the receiver. The first frame transmitted by the sender MAY set this field to 0.

3. Theory of Operations

This mechanism allows operator to measure the loss of BFD CC frames.

This measurement counts the number of BFD control frames missed at the receiver due to a transient change in the network such as congestion. Frame-loss is detected by comparing the Sequence Number field in the 0-Auth TLV in successive BFD CC frames. The Sequence

Number in each successive control frame generated on a BFD session by the transmitter is incremented by one.

The first BFD Loss-Delay TLV processed by the receiver that has a non-zero sequence number is used for bootstrapping the logic. Each successive frame after this is expected to have a Sequence Number that is one greater than the Sequence Number in the previous frame.

4. IANA Requirements

IANA is requested to assign new Auth-Type for the Null-Authentication TLV for BFD Stability Measurement. The following number is suggested.

Value Meaning

6 Null-Authentication TLV

5. Security Consideration

Since this method uses an authentication TLV to achive the functionality, usage of this TLV will prevent the use of other authentication TLVs.

6. Acknowledgements

Nobo Akiya, Jeffery Haas, Peng Fan, Dileep Singh, Basil Saji, Sagar Soni and Mallik Mudigonda also conributed to this document.

7. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, June 2010.

Authors' Addresses

Ashesh Mishra
Ciena Corporation
3939 North 1st Street
San Jose, CA 95134
USA

Email: mishra.ashesh@gmail.com
URI: www.ciena.com

Mahesh Jethanandani
Ciena Corporation
3939 North 1st Street
San Jose, CA 95134
USA

Email: mjethanandani@gmail.com
URI: www.ciena.com

Ankur Saxena
Ciena Corporation
3939 North 1st Street
San Jose, CA 95134
USA

Email: ankurpsaxena@gmail.com

Santosh Pallagatti
Juniper Networks
Juniper Networks, Exora Business Park
Bangalore, Karnataka 560103
India

Phone: +
Email: santoshpk@juniper.net

Mach Chen
Huawei

Email: mach.chen@huawei.com

Internet Engineering Task Force
Internet-Draft
Updates: 5885 (if approved)
Intended status: Standards Track
Expires: September 07, 2015

V. Govindan
C. Pignataro
Cisco Systems

March 06, 2015

Seamless BFD for VCCV
draft-gp-pals-seamless-vccv-00

Abstract

This document extends the procedures and Connectivity Verification (CV) types already defined for Bidirectional Forwarding Detection (BFD) for Virtual Circuit Connectivity Verification (VCCV) to define Seamless BFD (S-BFD) for VCCV. This document will be extended in future to include definition of procedures for S-BFD over Tunnels. This document extends the CV values defined in RFC5885.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on September 07, 2015.

Copyright Notice

Table of Contents

1. Background

BFD for VCCV [RFC5885] defines the CV types for BFD using VCCV, protocol operation and the required packet encapsulation formats. This document extends those procedures, CV type values to enable S-BFD [I-D.ietf-bfd-seamless-base] operation for VCCV.

The new S-BFD CV Types are PW demultiplexer-agnostic, and hence applicable for both MPLS and Layer Two Tunneling Protocol version 3 (L2TPv3) pseudowire demultiplexers. This document concerns itself with the S-BFD VCCV operation over single-segment pseudowires (SS-PWs). The scope of this document is as follows:

This specification describes procedures only for S-BFD asynchronous mode.

S-BFD Echo mode is outside the scope of this specification.

S-BFD operation for fault detection and status signaling is outside the scope of this specification.

1.1. 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 [RFC2119].

2. S-BFD Connectivity Verification

S-BFD protocol provides continuity check services by monitoring the S-BFD control packets sent and received over the VCCV channel of the PW. The term <Connectivity Verification> is used throughout this document to be consistent with [RFC5885].

This section defines the CV types to be used for S-BFD. It also defines the procedures for S-BFD discriminator advertisement for the SBD reflector and the procedure for S-BFD Initiator operation.

Two CV Types are defined for S-BFD. Table 1 summarizes the S-BFD CV Types, grouping them by encapsulation (i.e., with versus without IP/UDP headers) for fault detection only. S-BFD for fault detection and status signaling is outside the scope of this specification.

	Fault Detection Only	Fault Detection and Status Signaling
S-BFD, IP/UDP Encapsulation (with IP/UDP Headers)	TBD1 (Note1)	N/A
S-BFD, PW-ACH Encapsulation when using MPLS PW or L2SS Encapsulation when using L2TP	TBD2 (Note2)	N/A

| PW (without IP/UDP Headers) | | |
+---+---+---+

Table 1: Bitmask Values for BFD CV Types

Two new bits are requested from IANA to indicate S-BFD operation.

2.1. Co-existence of S-BFD and BFD capabilites

Since the CV types for S-BFD and BFD are unique, BFD and S-BFD capabilities can be advertised concurrently.

2.2. S-BFD CV Operation

2.2.1. S-BFD Initiator Operation

The S-BFD Initiator SHOULD bootstrap S-BFD sessions after it learns the discriminator of the remote target identifier through one or more of the following methods:

1. Advertisements of S-BFD discriminators made through AVP/ TLVs defined in L2TP/ LDP.

2. Provisioning of S-BFD discriminators.

3. Probing remote S-BFD discriminators through S-BFD Alert discriminators [I-D.akiya-bfd-seamless-alert-discrim]

S-BFD Initiator operation MUST be according to the specifications in Section 7.2 of [I-D.ietf-bfd-seamless-base].

2.2.2. S-BFD Reflector Operation

When as pseudowire signalling protocol such as LDP or L2TPv3 is in use the S-BFD Reflector advertises its target discriminators using that signalling protocol. When static PWs are in use the target discriminator of S-BFD needs to be provisioned on the S-BFD Initiator nodes.

All point to point pseudowires are bidirectional, the S-BFD Reflector therefore reflects the S-BFD packet back to the Initiator using the VCCV channel of the reverse direction of the PW on which it was received.

2.2.2.1. S-BFD Reflector Demultiplexing

TBD

2.2.2.2. S-BFD Reflector transmission of control packets

The procedures of S-BFD Reflector described in [I-D.ietf-bfd-seamless-base] apply for S-BFD using VCCV.

2.2.2.3. S-BFD Reflector advertisement of target discriminators using LDP

TBD.

2.2.2.4. S-BFD Reflector advertisement of target discriminators using L2TP

The S-BFD Reflector MUST use the AVP [I-D.gp-l2tpext-sbfd-discriminator] defined for advertising its target discriminators using L2TP.

2.2.2.5. Provisioning of S-BFD Reflector target discriminators

S-BFD target discriminators MAY be provisioned when static PWs are used.

2.2.2.6. Probing of S-BFD Reflector target discriminators using alert discriminators

S-BFD alert discriminators MAY be used to probe S-BFD target discriminators. If a node implements S-BFD reflector, it SHOULD respond to Alert discriminator requests received from potential S-BFD Initiators.

2.3. S-BFD Encapsulation

Unless specified differently below, the encapsulation of S-BFD packets is the identical the method specified in Sec.3.2 [RFC5885] and in [RFC5880] for the encapsulation of BFD packets.

- IP/UDP BFD Encapsulation (BFD with IP/UDP Headers)

  The destination UDP port for the IP encapsulated S-BFD packet MUST be 7784 [I-D.ietf-bfd-seamless-base].

  The encapsulation of the S-BFD header fields MUST be according to Sec.7.2.2 of [I-D.ietf-bfd-seamless-base].

- PW-ACH/ L2SS BFD Encapsulation (BFD without IP/UDP Headers)

The encapsulation of S-BFD packets using this format MUST be according to Sec.3.2 of [RFC5885] with the exception of the PW-ACH/ L2SS type.

When VCCV carries PW-ACH/ L2SS-encapsulated S-BFD (i.e., "raw" S-BFD), the PW-ACH (pseudowire CW's) or L2SS' Channel Type MUST be set to TBD2 to indicate "S-BFD Control, PW-ACH/ L2SS-encapsulated" (i.e., S-BFD without IP/UDP headers; see Section 5.3). This is to allow the identification of the encased S-BFD payload when demultiplexing the VCCV control channel.

2.4. S-BFD CV Types

3. Capability Selection

When multiple S-BFD CV Types are advertised, and after applying the rules in [RFC5885], the set that both ends of the pseudowire have in common is determined. If the two ends have more than one S-BFD CV Type in common, the following list of S-BFD CV Types is considered in the order of the lowest list number CV Type to the highest list number CV Type, and the CV Type with the lowest list number is used:

1. TBD1 - S-BFD IP/UDP-encapsulated, for PW Fault Detection only.

2. TBD2 - S-BFD PW-ACH/ L2SS-encapsulated (without IP/UDP headers), for PW Fault Detection only.

The order of capability selection between S-BFD and BFD is TBD.

4. Security Considerations

Security measures described in [RFC5885] and [I-D.ietf-bfd-seamless-base] are to be followed.

5. IANA Considerations

5.1. MPLS CV Types for the VCCV Interface Parameters Sub-TLV

The VCCV Interface Parameters Sub-TLV codepoint is defined in [RFC4446], and the VCCV CV Types registry is defined in [RFC5085].

This section lists the new BFD CV Types.

IANA has augmented the "VCCV Connectivity Verification (CV) Types" registry in the Pseudowire Name Spaces reachable from [IANA]. These are bitfield values. CV Type values TBD are specified in Section 2 of this document.

MPLS Connectivity Verification (CV) Types:

Bit (Value)	Description	Reference
TBD1(0xY)	S-BFD IP/UDP-encapsulated, for PW Fault Detection only	this document
TBD2(0xZ)	S-BFD PW-ACH/L2SS-encapsulated, for PW Fault Detection only	this document

5.2. L2TPv3 CV Types for the VCCV Capability AVP

This section lists the new requests for S-BFD CV Types to be added to the existing "VCCV Capability AVP" registry in the L2TP name spaces. The Layer Two Tunneling Protocol "L2TP" Name Spaces are reachable from [IANA]. IANA is requested to assign the following L2TPv3 Connectivity Verification (CV) Types in the VCCV Capability AVP Values registry.

VCCV Capability AVP (Attribute Type 96) Values

L2TPv3 Connectivity Verification (CV) Types:

Bit (Value)	Description	Reference
TBD1(0xY)	S-BFD IP/UDP-encapsulated, for PW Fault Detection only	this document
TBD2(0xZ)	S-BFD L2SS-encapsulated, for PW Fault Detection only	this document

5.3. PW Associated Channel Type

As per the IANA considerations in [RFC5586], IANA is requested to allocate the following Channel Types in the "MPLS Generalized Associated Channel (G-ACh) Types" registry:

IANA has reserved a new Pseudowire Associated Channel Type value as follows:

Registry:

Value	Description	TLV Follows	Reference
TBD2	S-BFD Control, PW-ACH/L2SS encapsulation (without IP/UDP Headers)	No	[This document]

6. Acknowledgements

Authors would like to thank Nobo Akiya, Stewart Bryant and Pawel Sowinski for providing the core inputs of this document and for performing thorough reviews and providing number of comments.

7. Contributing Authors

Mallik Mudigonda
Cisco Systems
Email: mmudigon@cisco.com

8. References

8.1. Normative References

[I-D.akiya-bfd-seamless-alert-discrim]
Akiya, N., Pignataro, C., and D. Ward, "Seamless Bidirectional Forwarding Detection (S-BFD) Alert Discriminator", draft-akiya-bfd-seamless-alert-discrim-03 (work in progress), October 2014.

[I-D.gp-l2tpext-sbfd-discriminator]
Govindan, V. and C. Pignataro, "Advertising S-BFD Discriminators in L2TPv3", draft-gp-l2tpext-sbfd-discriminator-00 (work in progress), March 2015.

[I-D.ietf-bfd-seamless-base]
Akiya, N., Pignataro, C., Ward, D., Bhatia, M., and J. Networks, "Seamless Bidirectional Forwarding Detection (S-BFD)", draft-ietf-bfd-seamless-base-04 (work in progress), January 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, February 2006.

[RFC4446] Martini, L., "IANA Allocations for Pseudowire Edge to Edge Emulation (PWE3)", BCP 116, RFC 4446, April 2006.

[RFC5085] Nadeau, T. and C. Pignataro, "Pseudowire Virtual Circuit Connectivity Verification (VCCV): A Control Channel for Pseudowires", RFC 5085, December 2007.

[RFC5586] Bocci, M., Vigoureux, M., and S. Bryant, "MPLS Generic Associated Channel", RFC 5586, June 2009.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, June 2010.

[RFC5885] Nadeau, T. and C. Pignataro, "Bidirectional Forwarding Detection (BFD) for the Pseudowire Virtual Circuit Connectivity Verification (VCCV)", RFC 5885, June 2010.

8.2. Informative References

[IANA] Internet Assigned Numbers Authority , "Protocol Registries", , <http://www.iana.org>.

Authors' Addresses

Vengada Prasad Govindan
Cisco Systems

Email: venggovi@cisco.com

Carlos Pignataro
Cisco Systems

Email: cpignata@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 17, 2015

M. Jethanandani
A. Mishra
A. Saxena
Ciena Corporation
M. Bhatia
Ionos Networks

February 13, 2015

Optimizing BFD Authentication
draft-mahesh-bfd-authentication-00

Abstract

This document describes an optimization to BFD Authentication as described in Section 6.7 of BFD [RFC5880].

Requirements Language

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 RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on August 17, 2015.

Copyright Notice

Table of Contents

1. Introduction

Authenticating every BFD [RFC5880] packet with a Simple Password, or with a MD5 Message-Digest Algorithm [RFC1321] and Secure Hash Algorithm (SHA-1) algorithms is computationally intensive process, making it difficult if not impossible to authenticate every packet - particularly at faster intervals. In addition, the recent escalating series of attacks on MD5 and SHA-1 [SHA-1-attack1] [SHA-1-attack2] raise concerns about their remaining useful lifetime as outlined in Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithm [RFC6151] and Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithm [RFC6194]. If replaced by stronger algorithms, the computational requirement of a stronger algorithms will make the task of authenticating every packet even more difficult to achieve.

This document proposes that only BFD frames that signal a state change in BFD be authenticated. The rest of the frames can be transmitted and received without authentication enabled. Bulk of the frames that are transmitted and received have no state change associated with them. Limiting authentication to frames that affect a BFD session state allows for more sessions to be supported for authentication. Moreover, most BFD frames that signal a state change are generally transmitted at a slower interval of 1s leaving enough time to compute the hash.

Section 2 talks about the changes to authentication mode as described in BFD [RFC5880].

2. Authentication Mode

The cryptographic authentication mechanisms specified in BFD [RFC5880] describes enabling and disabling of authentication as a one time operation. As a security precaution, it mentions that authentication state be allowed to change at most once. Once turned on, the document talks about every packet being enabled with Authentication bit and payload. In addition, it states that an implementation SHOULD NOT allow the authentication state to be changed based on the receipt of a BFD Control packet.

This document proposes that the authentication mode be modified to be enabled on demand. Instead of every packet being authenticated, the two ends can decide which frames need to be authenticated, and authenticate only those frames. For example, the two ends can decide that BFD frames that indicate a state change should be authenticated and enable authentication on those frames only. If the two ends have not previously negotiated which frames they will transmit or receive with authentication enabled, then the BFD session will fail to come up, because at least one end will expect every frame to be authenticated.

3. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

4. Security Considerations

The approach described in this document enhances the ability to authentication a BFD session by taking away the onerous requirement that every frame be authenticated. By authenticating frames that affect the state of the session, the security of the BFD session is maintained. As such this document does not change the security considerations for BFD.

5. References

5.1. Normative References

[FIPS-180-2]
National Institute of Standards and Technology, FIPS PUB 180-2, "The Keyed-Hash Message Authentication Code (HMAC)", August 2002.

[FIPS-198]
National Institute of Standards and Technology, FIPS PUB 198, "The Keyed-Hash Message Authentication Code (HMAC)", March 2002.

[I-D.ietf-bfd-generic-crypto-auth]
Bhatia, M., Manral, V., Zhang, D., and M. Jethanandani, "BFD Generic Cryptographic Authentication", draft-ietf-bfd-generic-crypto-auth-06 (work in progress), April 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6039] Manral, V., Bhatia, M., Jaeggli, J., and R. White, "Issues with Existing Cryptographic Protection Methods for Routing Protocols", RFC 6039, October 2010.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, March 2011.

[RFC6194] Polk, T., Chen, L., Turner, S., and P. Hoffman, "Security Considerations for the SHA-0 and SHA-1 Message-Digest Algorithms", RFC 6194, March 2011.

5.2. Informative References

[Dobb96a] Dobbertin, H., "Cryptanalysis of MD5 Compress", May 1996.

[Dobb96b] Dobbertin, H., "The Status of MD5 After a Recent Attack", CryptoBytes", 1996.

[I-D.ietf-karp-design-guide]
Lebovitz, G. and M. Bhatia, "Keying and Authentication for Routing Protocols (KARP) Design Guidelines", draft-ietf-karp-design-guide-10 (work in progress), December 2011.

[MD5-attack]
Wang, X., Feng, D., Lai, X., and H. Yu, "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD", August 2004.

[NIST-HMAC-SHA]
National Institute of Standards and Technology, Available online at http://csrc.nist.gov/groups/ST/hash/policy.html, "NIST’s Policy on Hash Functions", 2006.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4822] Atkinson, R. and M. Fanto, "RIPv2 Cryptographic Authentication", RFC 4822, February 2007.

[RFC5310] Bhatia, M., Manral, V., Li, T., Atkinson, R., White, R., and M. Fanto, "IS-IS Generic Cryptographic Authentication", RFC 5310, February 2009.

[RFC5709] Bhatia, M., Manral, V., Fanto, M., White, R., Barnes, M., Li, T., and R. Atkinson, "OSPFv2 HMAC-SHA Cryptographic Authentication", RFC 5709, October 2009.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, June 2010.

[RFC6234] Eastlake, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, May 2011.

[SHA-1-attack1] Wang, X., Yin, Y., and H. Yu, "Finding Collisions in the Full SHA-1", 2005.

[SHA-1-attack2] Wang, X., Yao, A., and F. Yao, "New Collision Search for SHA-1", 2005.

Authors' Addresses

Mahesh Jethanandani
Ciena Corporation
3939 North 1st Street
San Jose, CA 95134
USA

Phone: +1 (408) 904-2160
Email: mjethanandani@gmail.com

Ashesh Mishra
Ciena Corporation
3939 North 1st Street
San Jose, CA 95134
USA

Phone: +1 (408) 904-2114
Email: mishra.ashesh@gmail.com

Ankur Saxena
Ciena Corporation
3939 North 1st Street
San Jose, CA 95134
USA

Email: ankurpsaxena@gmail.com

Manav Bhatia
Ionos Networks
Bangalore
India

Email: manav@ionosnetworks.com

MPLS Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2015

G. Mirsky
J. Tantsura
Ericsson
I. Varlashkin
Google
M. Chen
Huawei

March 5, 2015

Bidirectional Forwarding Detection (BFD) Directed Return Path
draft-mirsky-mpls-bfd-directed-03

Abstract

Bidirectional Forwarding Detection (BFD) is expected to monitor bi-directional paths. When a BFD session monitors in its forward direction an explicitly routed path there is a need to be able to direct egress BFD peer to use specific path as reverse direction of the BFD session.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on September 6, 2015.

Copyright Notice

Table of Contents

1. Introduction

RFC 5880 [RFC5880], RFC 5881 [RFC5881], and RFC 5883 [RFC5883] established the BFD protocol for IP networks and RFC 5884 [RFC5884] set rules of using BFD asynchronous mode over IP/MPLS LSPs. All standards implicitly assume that the egress BFD peer will use the shortest path route regardless of route being used to send BFD control packets towards it. As result, if the ingress BFD peer sends its BFD control packets over explicit path that is diverging from the best route, then reverse direction of the BFD session is likely not to be on co-routed bi-directional path with the forward direction of the BFD session. And because BFD control packets are not guaranteed to cross the same links and nodes in both directions detection of Loss of Continuity (LoC) defect in forward direction may demonstrate positive negatives.

This document defines the extension to LSP Ping [RFC4379], BFD Reverse Path TLV, and proposes that it to be used to instruct the

egress BFD peer to use explicit path for its BFD control packets associated with the particular BFD session. The TLV will be allocated from the TLV and sub-TLV registry defined by RFC 4379 [RFC4379]. As a special case, forward and reverse directions of the BFD session can form bi-directional co-routed associated channel.

1.1. Conventions used in this document

1.1.1. Terminology

BFD: Bidirectional Forwarding Detection

MPLS: Multiprotocol Label Switching

LSP: Label Switching Path

LoC: Loss of Continuity

1.1.2. 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 [RFC2119].

2. Problem Statement

BFD is best suited to monitor bi-directional co-routed paths. In most cases, given stable environments, the forward and reverse direction between two nodes is likely to be co-routed, this fulfilling the implicit BFD requirements. If BFD is used to monitor unidirectional explicitly routed paths, e.g. MPLS-TE LSPs, its control packets in forward direction would be in-band using the mechanism defined in [RFC5884] and [RFC5586]. But the reverse direction of the BFD session would still follow the shortest path route and that might lead to the following problems detecting failures on the unidirectional explicit path:

- failure detection on the reverse path cannot be interpreted as bi-directional failure and thus trigger, for example, protection switchover of the forward direction;
- if reverse direction is in Down state, the head-end node would not receive indication of forward direction failure from its egress peer.

To address these challenges the egress BFD peer should be instructed to use specific path for its control packets.

3. Direct Reverse BFD Path

3.1. Case of MPLS Data Plane

LSP ping, defined in [RFC4379], uses BFD Discriminator TLV [RFC5884] to bootstrap a BFD session over an MPLS LSP. This document defines a new TLV, BFD Reverse Path TLV, that MUST contain a single sub-TLV that can be used to carry information about reverse path for the specified in BFD Discriminator TLV session.

3.1.1. BFD Reverse Path TLV

The BFD Reverse Path TLV is an optional TLV within the LSP ping protocol. However, if used, the BFD Discriminator TLV MUST be included in an Echo Request message as well. If the BFD Discriminator TLV is not present when the BFD Reverse Path TLV is included, then it MUST be treated as malformed Echo Request, as described in [RFC4379].

The BFD Reverse Path TLV carries the specified path that BFD control packets of the BFD session referenced in the BFD Discriminator TLV are required to follow. The format of the BFD Reverse Path TLV is as presented in Figure 1.

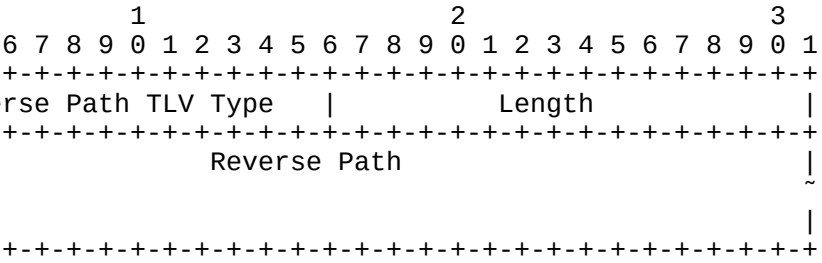


Figure 1: BFD Reverse Path TLV

BFD Reverse Path TLV Type is 2 octets in length and value to be assigned by IANA.

Length is 2 octets in length and defines the length in octets of the Reverse Path field.

Reverse Path field contains a sub-TLV. Any Target FEC sub-TLV, already or in the future defined, from IANA sub-registry Sub-TLVs for TLV Types 1, 16, and 21 of MPLS LSP Ping Parameters registry MAY be used in this field. Only one sub-TLV MUST be included in the Reverse Path TLV. If more than one sub-TLVs are present in the Reverse Path

TLV, then only the first sub-TLV MUST be used and the rest MUST be silently discarded.

If the egress LSR fails to establish the BFD session because path specified in the Reverse Path TLV is not known, the egress MAY establish the BFD session over IP network [RFC5884] and MAY send Echo Reply with the Reverse Path TLV received and the return code set to "Failed to establish the BFD session". The specified reverse path was not found" (TBD4) Section 3.4. If the egress LSR cannot find path specified in the Reverse Path TLV and does not establish BFD session per RFC 5884, it MUST send Echo Reply with the Reverse Path TLV received and the return code set to "Failed to establish the BFD session. The specified reverse path was not found".

3.1.2. Segment Routing Tunnel sub-TLV

With MPLS data plane explicit path can be either Static or RSVP-TE LSP, or Segment Routing tunnel. In case of Static or RSVP-TE LSP [RFC7110] defined sub-TLVs to identify explicit return path. For the Segment Routing with MPLS data plane case a new sub-TLV is defined in this document as presented in Figure 2.

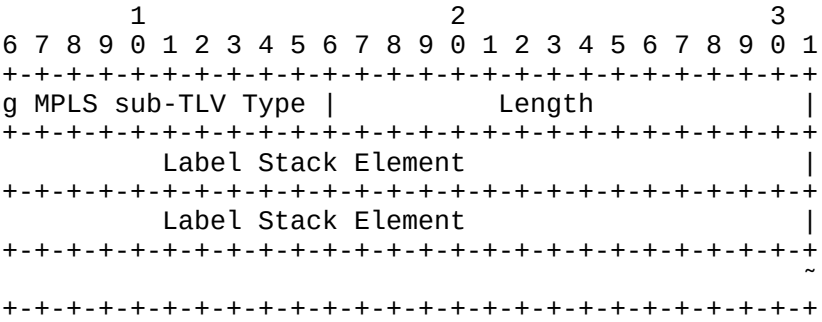


Figure 2: Segment Routing MPLS Tunnel sub-TLV

The Segment Routing Tunnel sub-TLV Type is two octets in length, and will be allocated by IANA.

The Segment Routing Tunnel sub-TLV MAY be used in Reply Path TLV defined in [RFC7110]

3.2. Case of IPv6 Data Plane

IPv6 can be data plane of choice for Segment Routed tunnels [I-D.previdi-6man-segment-routing-header]. In such networks the BFD Reverse Path TLV described in Section 3.1.1 can be used as well. IP networks, unlike IP/MPLS, do not require use of LSP ping with BFD

Discriminator TLV[RFC4379] to bootstrap BFD session. But to specify reverse path of a BFD session in IPv6 environment the BFD Discriminator TLV MUST be used along with the BFD Reverse Path TLV. The BFD Reverse Path TLV in IPv6 network MUST include sub-TLV.

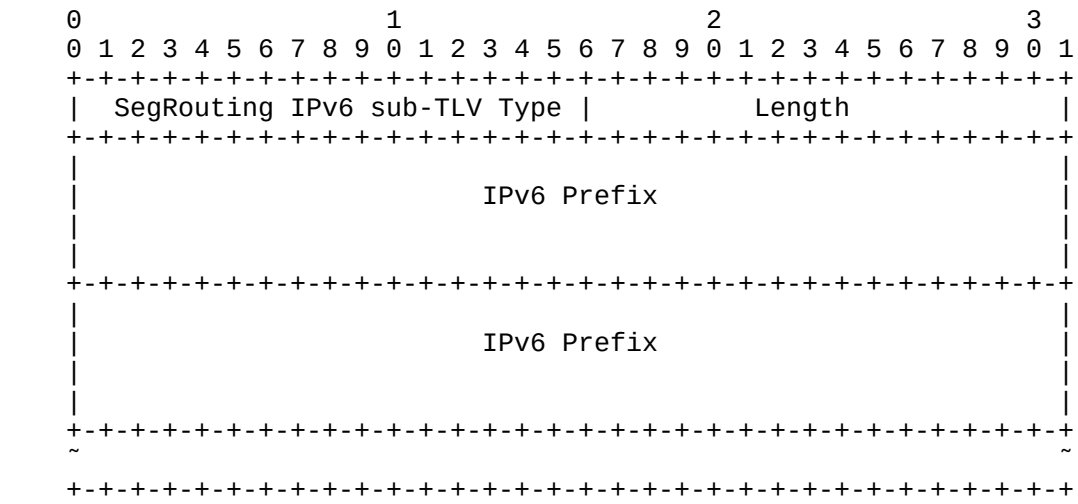


Figure 3: Segment Routing IPv6 Tunnel sub-TLV

3.3. Bootstrapping BFD session with BFD Reverse Path over Segment Routed tunnel

As discussed in [I-D.kumarkini-mpls-spring-lsp-ping] introduction of Segment Routing network domains with MPLS dataplane adds three new sub-TLVs that may be used with Target FEC TLV. Section 6.1 addresses use of new sub-TLVs in Target FEC TLV in LSP ping and LSP traceroute. For the case of LSP ping the [I-D.kumarkini-mpls-spring-lsp-ping] states that:

"Initiator MUST include FEC(s) corresponding to the destination segment.

Initiator MAY include FECs corresponding to some or all of segments imposed in the label stack by the initiator to communicate the segments traversed. "

When LSP ping is used to bootstrap BFD session this document updates this and defines that LSP Ping MUST include the FEC corresponding to the destination segment and SHOULD NOT include FECs corresponding to some or all of segment imposed by the initiator. Operationally such restriction would not cause any problem or uncertainty as LSP ping

with FECs corresponding to some or all segments or traceroute may preceed the LSP ping that bootstraps the BFD session.

3.4. Return Codes

This document defines the following Return Codes:

- Failed to establish the BFD session. The specified reverse path was not found, (TBD4) - the specified reverse path was not found, failed to establish the BFD session. When a specified reverse path is not available at the egress LSR, an Echo Reply with the return code set to "Failed to establish the BFD session. The specified reverse path was not found." MAY be sent back to the initiator . (Section 3.1.1)

4. Use Case Scenario

In network presented in Figure 4 node A monitors two tunnels to node H: A-B-C-D-G-H and A-B-E-F-G-H. To bootstrap BFD session to monitor the first tunnel, node A MUST include BFD Discriminator TLV with Discriminator value N and MAY include BFD Reverse Path TLV that references H-G-D-C-B-A tunnel. To bootstrap BFD session to monitor the second tunnel, node A MUST include BFD Discriminator TLV with Discriminator value M and MAY include BFD Reverse Path TLV that references H-G-F-E-B-A tunnel.

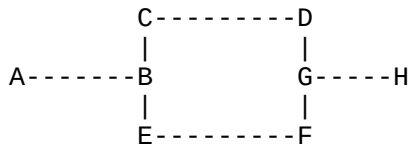


Figure 4: Use Case for BFD Reverse Path TLV

If an operator needs node H to monitor path to node A, e.g. H-G-D-C-B-A tunnel, then by looking up list of known Reverse Paths it MAY find and use existing BFD sessions.

5. IANA Considerations

5.1. TLV

The IANA is requested to assign a new value for BFD Reverse Path TLV from the "Multiprotocol Label Switching Architecture (MPLS) Label Switched Paths (LSPs) Ping Parameters - TLVs" registry, "TLVs and sub-TLVs" sub-registry.

Value	Description	Reference
X (TBD1)	BFD Reverse Path TLV	This document

Table 1: New BFD Reverse Type TLV

5.2. Sub-TLV

The IANA is requested to assign two new sub-TLV types from "Multiprotocol Label Switching Architecture (MPLS) Label Switched Paths (LSPs) Ping Parameters - TLVs" registry, "Sub-TLVs for TLV Types 1, 16, and 21" sub-registry.

Value	Description	Reference
X (TBD2)	Segment Routing MPLS Tunnel sub-TLV	This document
X (TBD3)	Segment Routing IPv6 Tunnel sub-TLV	This document

Table 2: New Segment Routing Tunnel sub-TLV

5.3. Return Codes

The IANA is requested to assign a new Return Code value from the "Multi-Protocol Label Switching (MPLS) Label Switched Paths (LSPs) Ping Parameters" registry, "Return Codes" sub-registry, as follows using a Standards Action value.

Value	Description	Reference
X (TBD4)	Failed to establish the BFD session. The specified reverse path was not found.	This document

Table 3: New Return Code

6. Security Considerations

Secuirity considerations discussed in [RFC5880], [RFC5884], and [RFC4379], apply to this document.

7. Acknowledgements

8. Normative References

[I-D.kumarkini-mpls-spring-lsp-ping]
Kumar, N., Swallow, G., Pignataro, C., Akiya, N., Kini, S., Gredler, H., and M. Chen, "Label Switched Path (LSP) Ping/Trace for Segment Routing Networks Using MPLS Dataplane", draft-kumarkini-mpls-spring-lsp-ping-02 (work in progress), October 2014.

[I-D.previdi-6man-segment-routing-header]
Previdi, S., Filsfils, C., Field, B., and I. Leung, "IPv6 Segment Routing Header (SRH)", draft-previdi-6man-segment-routing-header-05 (work in progress), January 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", RFC 4379, February 2006.

[RFC5586] Bocci, M., Vigoureux, M., and S. Bryant, "MPLS Generic Associated Channel", RFC 5586, June 2009.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, June 2010.

[RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, June 2010.

[RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths", RFC 5883, June 2010.

[RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", RFC 5884, June 2010.

[RFC7110] Chen, M., Cao, W., Ning, S., Jounay, F., and S. Delord, "Return Path Specified Label Switched Path (LSP) Ping", RFC 7110, January 2014.

Authors' Addresses

Greg Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com

Jeff Tantsura
Ericsson

Email: jeff.tantsura@ericsson.com

Ilya Varlashkin
Google

Email: Ilya@nobulus.com

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com

TRILL
Internet-Draft
Intended status: Standards Track
Expires: August 7, 2015

M. Zhang
Huawei Technologies
S. Pallagatti
Juniper Networks
V. Govindan
Cisco Systems

February 03, 2015

Point to Multipoint BFD for TRILL
draft-zhang-trill-p2mp-bfd-00

Abstract

Point to multipoint (P2MP) BFD is designed to verify multipoint connectivity. This document specifies the support of P2MP BFD in TRILL. Similar as TRILL point to point BFD, BFD Control packets in TRILL P2MP BFD are also transmitted using an extended RBridge Channel.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on August 7, 2015.

Table of Contents

1. Introduction

TRILL supports multicast forwarding. Applications based on TRILL multicast wish to achieve quick detection of multicast failures using P2MP BFD. This document specifies the use of P2MP BFD in TRILL.

To use P2MP BFD, the head need to periodically transmit BFD Control packets to all tails using TRILL multicast. A new RBridge Channel is allocated for this purpose.

In order to execute the global protction of distribution used for multicast forwarding [I-D.ietf-trill-resilient-trees], the head need to track the active status of tails [spallagatti-bfd-multipoint-active-tail]. When the tail loses connectivity from the head, it should notify the head of the lack of multipoint connectivity with unicast BFD Control packets. These packets are transmitted using the existing RBridge Channel assigned to BFD Control [RFC7175].

2. Acronyms and Terminology

2.1. Acronyms

Data Label: VLAN or Fine Grained Label [RFC7172].

BFD: Bidirectional Forwarding Detection

2.2. Terminology

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].

Familiarity with [RFC6325][RFC7175][RFC7178] is assumed in this document.

3. Bootstrapping

The TRILL adjacency mechanism bootstraps the establishment of the BFD session [RFC7177]. A slight wording update to the second sentence in Section 6 of [RFC7177] is required.

It currently read:

> If an RBridge supports BFD [RFC7175], it will have learned whether the other RBridge has BFD enabled by whether or not a BFD-Enabled TLV [RFC6213] was included in its Hellos.

Now it should read:

> If an RBridge supports BFD [RFC7175] [this document], it will have learned whether the other RBridge has BFD enabled by whether or not a BFD-Enabled TLV [RFC6213] was included in its Hellos.

4. A New RBridge Channel for P2MP BFD

RBridge Channel 0x002 is defined for TRILL point to point BFD Control packets in [RFC7175]. If the M bit of the TRILL Header of the channeled packet containing the BFD Control packet is non-zero, the packet MUST be dropped [RFC7175]. While for P2MP BFD, the head is required to probe tails using multicast. This means the M bit will be set to 1. For this reason, a new RBridge Channel, whose code point is TBD, is specified in this document. An RBridge that supports P2MP BFD MUST support the new RBridge Channel for P2MP BFD. The capability to support the RBridge Channel for P2MP BFD, and therefore support performing P2MP BFD, is announced within the "RBridge Channel Protocols Sub-TLV" in LSPs [RFC7176].

An alternative option is to define a new RBridge Channel Tunnel protocol for P2MP BFD Control packets [I-D.ietf-trill-channel-tunnel] so that P2MP BFD Control Packets can be adapted as the payload of this Tunnel protocol.

As specified in [RFC7178], when the tail receives TRILL Data packets sent on the channel, it will absorb the packets itself rather than deliever these packets to its attached end-stations.

5. Discriminators and Packet Demultiplexing

In [I-D.ietf-bfd-multipoint], the tail demultiplexes incoming BFD packets based on a combination of the source address and My Discriminator. In addition to this combination, TRILL P2MP BFD requires the tail to use the Data Label, which is either the inner VLAN or the Fine Grained Label [RFC7172], for demultiplexing. If the tail need to notify the head about the failure of a multipath, the tail is required to send unicast BFD Control packets using the same Data Label as used by the head.

6. Tracking Active Tails

According to[I-D.ietf-bfd-multipoint], the head has a session of type MultipointHead that is bound to a multipoint path. Multipoint BFD Control packets are sent by this session over the multipoint path, and no BFD Control packets are received by it. Each tail dynamically creates a MultipointTail per a multipoint path. MultipointTail sessions receive BFD Control packets from the head over multipoint paths.

If the head is keeping track of some or all of the tails [I-D.ietf-trill-resilient-trees], it has a session of type MultipointClient per tail that it cares about [spallagatti-bfd-multipoint-active-tail]. See [spallagatti-bfd-multipoint-active-tail] for detail operations of tacking active tails.

7. Security Considerations

P2MP BFD control packets can be encapsulated as the payload of the RBridge Channel Tunnel [I-D.ietf-trill-channel-tunnel]. In that case, the security option of RBridge Channel Tunnel can secure the transmission of BFD control packets.

The demultiplexing of TRILL P2MP BFD at the tail is Data Label aware. This enhances the security of the dynamic creation of MultipointTail sessions at tails. In order to forge BFD Control packets, the attacker has to acquire the right Data Label that the head uses for P2MP BFD.

8. IANA Considerations

IANA is required to allocate one RBridge Channel protocol number from the Standards Action range, as follows:

Protocol	Number
P2MP BFD Control	TBD

9. References

9.1. Normative References

[I-D.ietf-bfd-multipoint]
Katz, D., Ward, D., and J. Networks, "BFD for Multipoint Networks", draft-ietf-bfd-multipoint-05 (work in progress), January 2015.

[I-D.ietf-trill-channel-tunnel]
Eastlake, D. and L. Yizhou, "TRILL: RBridge Channel Tunnel Protocol", draft-ietf-trill-channel-tunnel-02 (work in progress), December 2014.

[I-D.ietf-trill-resilient-trees]
Zhang, M., Senevirathne, T., Pathangi, J., Banerjee, A., and A. Ghanwani, "TRILL Resilient Distribution Trees", draft-ietf-trill-resilient-trees-02 (work in progress), December 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6325] Perlman, R., Eastlake, D., Dutt, D., Gai, S., and A. Ghanwani, "Routing Bridges (RBridges): Base Protocol Specification", RFC 6325, July 2011.

[RFC7172] Eastlake, D., Zhang, M., Agarwal, P., Perlman, R., and D. Dutt, "Transparent Interconnection of Lots of Links (TRILL): Fine-Grained Labeling", RFC 7172, May 2014.

[RFC7175] Manral, V., Eastlake, D., Ward, D., and A. Banerjee, "Transparent Interconnection of Lots of Links (TRILL): Bidirectional Forwarding Detection (BFD) Support", RFC 7175, May 2014.

[RFC7176] Eastlake, 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.

[RFC7177] Eastlake, D., Perlman, R., Ghanwani, A., Yang, H., and V. Manral, "Transparent Interconnection of Lots of Links (TRILL): Adjacency", RFC 7177, May 2014.

[RFC7178] Eastlake, D., Manral, V., Li, Y., Aldrin, S., and D. Ward, "Transparent Interconnection of Lots of Links (TRILL): RBridge Channel Support", RFC 7178, May 2014.

[spallagatti-bfd-multipoint-active-tail] Katz, D., Ward, D., and S. Pallagatti , "BFD Multipoint Active Tails", January 2015.

9.2. Informative References

[RFC6213] Hopps, C. and L. Ginsberg, "IS-IS BFD-Enabled TLV", RFC 6213, April 2011.

Authors' Addresses

Mingui Zhang
Huawei Technologies
No.156 Beiqing Rd. Haidian District
Beijing 100095
P.R. China

Email: zhangmingui@huawei.com

Santosh Pallagatti
Juniper Networks
Embassy Business Park
Bangalore KA 560093
India

Email: santoshpk@juniper.net

Vengada Prasad Govindan
Cisco Systems

Email: venggovi@cisco.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 7, 2015

L. Zheng, Ed.
Huawei Technologies
R. Rahman, Ed.
Cisco Systems
S. Pallagatti
Juniper Networks
M. Jethanandani
Ciena Corporation

March 6, 2015

Yang Data Model for Bidirectional Forwarding Detection (BFD)
draft-zheng-bfd-yang-00.txt

Abstract

This document defines a YANG data model that can be used to configure and manage Bidirectional Forwarding Detection (BFD).

Requirements Language

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 RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on September 7, 2015.

Copyright Notice

Table of Contents

1. Introduction

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g RESTCONF [I-D.ietf-netconf-restconf]) and encodings other than XML (e.g JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interfaces, such as CLI and programmatic APIs.

This document defines a YANG data model that can be used to configure and manage Bidirectional Forwarding Detection (BFD)[RFC5880]. BFD is a network protocol which is used for liveness detection of arbitrary paths between systems. Some examples of different types of paths over which we have BFD:

1) Two systems directly connected via IP. This is known as BFD over single-hop IP [RFC5881]

2) Two systems connected via multiple hops [RFC5883]

3) Two systems connected via MPLS Label Switched Paths (LSPs) [RFC5884]

4) Two systems connected via a Link Aggregation Group (LAG) interface [RFC7130]

BFD typically does not operate on its own. Various control protocols, aka BFD clients, use the services provided by BFD for their own operation [RFC5882]. The obvious candidates which use BFD are those which do not have Hellos to detect failures (e.g. static routes) and routing protocols whose Hellos do not suport sub-second failure detection, e.g OSPF and IS-IS.

1.1. Contributors

2. Design of the Data Model

2.1. Design of configuration model

The configuration model consists mainly of the parameters specified in [RFC5880]. Some examples are desired minimum transmit interval, required minimum receive interval, detection multiplier etc

Some implementations have BFD configuration under the BFD client, e.g. BFD configuration is under routing applications such as OSPF, IS-IS, BGP etc. Other implementations have BFD configuration

centralized, i.e outside the multiple BFD clients. In the sections below we address both approaches.

2.1.1. Centralized BFD configuration

The BFD data model consists of configuring BFD sessions of different types (e.g. single-hop IP, multi-hop IP etc). Since the different session types have different keys we have a list per session type, but we use a grouping to share the common configuration data between the different session types.

2.1.1.1. Common BFD configuration

The common BFD session configuration items are put in a grouping to be used in multiple places, these items are:

local-multiplier
: This is the detection time multiplier as defined in [RFC5880].

desired-min-tx-interval
: This is the Desired Min TX Interval as defined in [RFC5880].

required-min-rx-interval
: This is the Required Min RX Interval as defined in [RFC5880].

demand-enabled
: Set to True to enable demand mode as defined in [RFC5880].

enable-authentication
: Set to True to enable BFD authentication.

authentication-algorithm
: Authentication algorithm to use (if enabled).

key-chain-name
: Key-chain to be used for authentication (if enabled).

2.1.1.2. Single-hop IP

We have a list for BFD sessions over single-hop IP. The key consists of:

interface
: This is the interface on which the BFD packets for this session are transmitted and received. Examples of an interface are physical media, virtual circuit, tunnel etc.

destination address
Address belonging to the peer system as per [RFC5881]

The common configuration data in Section 2.1.1.1 is used for single-hop IP. On top of that common data, we also need configuration data for echo:

desired-min-echo-tx-interval
This is the minimum interval that the local system would like to use when transmitting BFD echo packets. If 0 the echo function as defined in [RFC5880] is disabled.

required-min-echo-rx-interval
The is the Required Min Echo RX Interval as defined in [RFC5880].

2.1.1.3. Multi-hop IP

We have a list for BFD sessions over multi-hop IP. The key consists of:

source address
Address belonging to the local system as per [RFC5883]

destination address
Address belonging to the remote system as per [RFC5883]

VRF name
VRF in which the BFD multi-hop session is running

The common configuration data in Section 2.1.1.1 is used for multi-hop IP.On top of that common data, we also need TTL:

ttl
TTL of outgoing BFD control packets.

2.1.1.4. MPLS LSP

TBD

2.1.1.5. Link Aggregation Group

TBD

2.1.1.6. Per-interface configuration

For implementations which have multiplier and intervals configured under the BFD clients we still need a central location to configure authentication, demand mode etc. This can be done by configuring the following parameters per interface:

Common parameters
: The common BFD parameters listed in Section 2.1.1.1

Echo parameters
: The echo parameters listed in Section 2.1.1.2

2.1.2. Configuration in BFD clients

When BFD is configured in BFD clients, it is highly desirable to have BFD configuration consistency between those clients. In this approach we have a grouping for BFD configuration which applications can import in their YANG module:

- This provides consistency since the same grouping is being used in all applications making use of BFD

- Since not all implementations of those BFD clients have support for BFD, we must use if-feature in the respective YANG modules

An application importing the BFD configuration grouping could do so in a hierarchical manner if it has multiple levels at which BFD configuration can be applied. In a subsequent section we provide an example of how a BFD client would use the grouping in such a way.

The configuration items are:

enabled
: Set to True to enable BFD.

local-multiplier
: This the detection time multiplier as defined in [RFC5880].

desired-min-tx-interval
: This the Desired Min TX Interval as defined in [RFC5880].

required-min-rx-interval
: This the Required Min RX Interval as defined in [RFC5880].

2.2. Design of operational model

The operational model contains both the overall statistics of BFD sessions running on the device and the per session operational statistics. Since BFD is used for liveness detection of arbitrary paths, there is no uniform key to identify a BFD session. e.g. a BFD single-hop IP session is uniquely identified by the combination of destination IP address and interface whereas a multihop IP session is uniquely identified by the combination of source IP address, destination IP address and VRF. For this reason, for per session operational statistics, we do not have a single list with different type BFD sessions. Instead we have a container in which we have multiple lists, where each list corresponds to one specific path type for BFD. For example we have one operational list for BFD single-hop IP, another list for BFD multi-hop IP etc. In each list, mainly three categories of operational items are shown. The fundamental information of a BFD session such as the local discriminator, remote discriminator and the capability of supporting demand detect mode are shown in the first category. A second category includes a BFD session running information, e.g. the FSM the device in and diagnostic code received. Another example is the actual transmit interval between the control packets, which may be different from the desired minimum transmit interval configured, is shown in this category. Similar examples are actual received interval between the control packets and the actual transmit interval between the echo packets. The third category contains the detailed statistics of this session, e.g. when the session went to up/down, how long it has been since the session is up/down.

2.3. Notifications

This YANG model defines a list of notifications to inform clients of BFD with important events detected during the protocol operation. Pair of local and remote discriminator identifies a BFD session on local system. Notification also give more important details about BFD sessions e.g. new state, time in previous state, VRF and reason for BFD session state changed.

2.4. RPC Operations

TBD

2.5. BFD Configuration Data Hierarchy

2.5.1. Centralized BFD configuration

The following is the centralized configuration data hierarchy:

We have a container which contains a list for each session type

We have per-interface configuration

```
module: bfd
   +--rw bfd-cfg
   |  +--rw bfd-session-cfg {bfd-centralized-session-config}?
   |  |  +--rw session-ip-sh* [interface dest-addr]
   |  |  |  +--rw interface                        if:interface-ref
   |  |  |  +--rw dest-addr                        inet:ip-address
   |  |  |  +--rw admin-down?                      boolean
   |  |  |  +--rw local-multiplier?                multiplier
   |  |  |  +--rw desired-min-tx-interval          uint32
   |  |  |  +--rw required-min-rx-interval         uint32
   |  |  |  +--rw demand-enabled?                  boolean
   |  |  |  +--rw enable-authentication?           boolean
   |  |  |  +--rw authentication-parms {bfd-authentication}?
   |  |  |  |  +--rw key-chain-name?   string
   |  |  |  |  +--rw algorithm?        bfd-auth-algorithm
   |  |  |  +--rw desired-min-echo-tx-interval?    uint32
   |  |  |  +--rw required-min-echo-rx-interval?   uint32
   |  |  +--rw session-ip-mh* [vrf-name source-addr dest-addr]
   |  |     +--rw vrf-name                    vrfName
   |  |     +--rw source-addr                 inet:ip-address
   |  |     +--rw dest-addr                   inet:ip-address
   |  |     +--rw admin-down?                 boolean
   |  |     +--rw local-multiplier?           multiplier
   |  |     +--rw desired-min-tx-interval     uint32
   |  |     +--rw required-min-rx-interval    uint32
   |  |     +--rw demand-enabled?             boolean
   |  |     +--rw enable-authentication?      boolean
   |  |     +--rw authentication-parms {bfd-authentication}?
   |  |     |  +--rw key-chain-name?   string
   |  |     |  +--rw algorithm?        bfd-auth-algorithm
   |  |     +--rw tx-ttl?                     TTL
   |  |     +--rw rx-ttl                      TTL
   |  +--rw bfd-interface-cfg* [interface] {bfd-interface-config}?
   |     +--rw interface                        if:interface-ref
   |     +--rw local-multiplier?                multiplier
   |     +--rw desired-min-tx-interval          uint32
   |     +--rw required-min-rx-interval         uint32
   |     +--rw demand-enabled?                  boolean
   |     +--rw enable-authentication?           boolean
   |     +--rw authentication-parms {bfd-authentication}?
   |     |  +--rw key-chain-name?   string
   |     |  +--rw algorithm?        bfd-auth-algorithm
   |     +--rw desired-min-echo-tx-interval?    uint32
   |     +--rw required-min-echo-rx-interval?   uint32
```

2.5.2. Configuration in BFD clients

The following is the configuration data hierarchy for a hypothetical BFD client called bfd-routing-app, the BFD configuration is supported conditionally via use of if-feature.

We have a list of areas and in each area we have a list of interfaces. The BFD configuration grouping is used in a hierarchical fashion, it can be applied in "area" and "interface":

- If BFD configuration is applied under an interface, that configuration takes precedence over any BFD configuration (if any) at the area level

- If BFD configuration is applied under an "area" and none of the interfaces in that area has BFD configuration, then all interfaces belong to the "area" in question inherit the BFD configuration for the area in question.

- If the BFD client implementation supports "interface all", then all the interfaces belonging to that area will inherit the BFD configuration under "interface all". Along with this if there are specific interface configuration then specific interface will override the "interface all" parameters.

```
module: bfd-routing-app
   +--rw area* [area-id]
      +--rw area-id      uint32
      +--rw bfd-cfg
      |  +--rw enabled?                    boolean
      |  +--rw local-multiplier?           multiplier
      |  +--rw desired-min-tx-interval     uint32
      |  +--rw required-min-rx-interval    uint32
      +--rw interface* [interface]
         +--rw interface    if:interface-ref
         +--rw bfd-cfg
            +--rw enabled?                    boolean
            +--rw local-multiplier?           multiplier
            +--rw desired-min-tx-interval     uint32
            +--rw required-min-rx-interval    uint32
```

2.6. Operational Data Hierarchy

The complete data hierarchy of BFD YANG operational model is presented below.

```
module: bfd
   +--rw bfd-oper
      +--ro bfd-session-statistics
      |  +--ro ip-sh-session-num?   uint32
      |  +--ro ip-mh-session-num?   uint32
      |  +--ro total-session-num?   uint32
      |  +--ro session-up-num?      uint32
      |  +--ro sess-down-num?       uint32
      +--ro bfd-session-lists
         +--ro session-ip-sh* [interface dest-addr]
         |  +--ro interface               if:interface-ref
         |  +--ro dest-addr               inet:ip-address
         |  +--ro sesssion-type?          enumeration
         |  +--ro local-discriminator?    discriminator
         |  +--ro remote-discriminator?   discriminator
         |  +--ro remote-multiplier?      multiplier
         |  +--ro out-interface?          if:interface-ref
         |  +--ro demand-capability?      boolean
         |  +--ro session-running*
         |  |  +--ro local-state?              state
         |  |  +--ro remote-state?             state
         |  |  +--ro local-diagnostic?         diagnostic
         |  |  +--ro remote-diagnostic?        diagnostic
         |  |  +--ro detect-Mode?              enumeration
         |  |  +--ro actual-tx-interval?       uint32
         |  |  +--ro actual-rx-interval?       uint32
         |  |  +--ro actual-echo-tx-interval?  uint32
         |  |  +--ro detect-time?              uint32
         |  +--ro sesssion-statistics*
         |     +--ro create-time?         yang:date-and-time
         |     +--ro last-down-time?      yang:date-and-time
         |     +--ro last-up-time?        yang:date-and-time
         |     +--ro receive-pkt?         uint64
         |     +--ro send-pkt?            uint64
         |     +--ro down-count?          uint32
         |     +--ro receive-bad-pkt?     uint64
         |     +--ro send-failed-pkt?     uint64
         |     +--ro short-break-count?   uint32
         +--ro session-ip-mh* [vrfName source-addr dest-addr]
            +--ro vrfName                  vrfName
            +--ro source-addr              inet:ip-address
            +--ro dest-addr                inet:ip-address
            +--ro ttl?                     TTL
            +--ro sesssion-type?           enumeration
            +--ro local-discriminator?     discriminator
            +--ro remote-discriminator?    discriminator
            +--ro remote-multiplier?       multiplier
            +--ro out-interface?           if:interface-ref
```

```
               +--ro demand-capability?      boolean
               +--ro session-running*
               |  +--ro local-state?              state
               |  +--ro remote-state?             state
               |  +--ro local-diagnostic?         diagnostic
               |  +--ro remote-diagnostic?        diagnostic
               |  +--ro detect-Mode?              enumeration
               |  +--ro actual-tx-interval?       uint32
               |  +--ro actual-rx-interval?       uint32
               |  +--ro actual-echo-tx-interval?  uint32
               |  +--ro detect-time?              uint32
               +--ro sesssion-statistics*
                  +--ro create-time?         yang:date-and-time
                  +--ro last-down-time?      yang:date-and-time
                  +--ro last-up-time?        yang:date-and-time
                  +--ro receive-pkt?         uint64
                  +--ro send-pkt?            uint64
                  +--ro down-count?          uint32
                  +--ro receive-bad-pkt?     uint64
                  +--ro send-failed-pkt?     uint64
                  +--ro short-break-count?   uint32
```

2.7. Notifications

The BFD YANG data model defines notifications for BFD session state changes.

```
module: bfd
notifications:
   +---n bfd-singlehop-notification
   |  +--ro local-discr?              discriminator
   |  +--ro remote-discr?             discriminator
   |  +--ro new-state?                state
   |  +--ro state-change-reason?      string
   |  +--ro time-in-previous-state?   string
   |  +--ro dest-addr?                inet:ip-address
   |  +--ro interface?                if:interface-ref
   |  +--ro echo-enabled?             boolean
   +---n bfd-multihop-notification
      +--ro local-discr?              discriminator
      +--ro remote-discr?             discriminator
      +--ro new-state?                state
      +--ro state-change-reason?      string
      +--ro time-in-previous-state?   string
      +--ro dest-addr?                inet:ip-address
      +--ro vrf-name?                 vrfName
      +--ro source-addr?              inet:ip-address
```

2.8. Examples

2.9. Interaction with other YANG modules

TBD.

2.10. BFD Yang Module

```
<CODE BEGINS>
module bfd {
  namespace "urn:ietf:params:xml:ns:yang:bfd";
  // replace with IANA namespace when assigned
  prefix "bfd";

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-inet-types {
    prefix "inet";
  }

  import ietf-yang-types {
    prefix yang;
  }

  organization "IETF BFD Working Group";

  contact
    "WG Web:   <http://tools.ietf.org/wg/bfd>
     WG List:  <rtg-bfd@ietf.org>
     WG Chair: Jeff Haas
     WG Chair: Nobo Akiya
     Editor:   Lianshu Zheng and Reshad Rahman";

  description
    "This module contains the YANG definition for BFD parameters as
     per RFC5880, RFC5881 and RFC5883";

  revision 2015-03-05 {
    description "Initial revision.";
  }

  typedef discriminator {
    type uint32 {
      range 1..4294967295;
    }
  }
```

```
  typedef diagnostic {
    type enumeration {
      enum none {
        value 0;
      }
      enum controlExpiry {
        value 1;
      }
      enum echoFailed {
        value 2;
      }
      enum nborDown {
        value 3;
      }
      enum fwdingReset {
        value 4;
      }
      enum pathDown {
        value 5;
      }
      enum concPathDown {
        value 6;
      }
      enum adminDown {
        value 7;
      }
      enum reverseConcPathDown {
        value 8;
      }
    }
  }

  typedef state {
    type enumeration {
      enum adminDown {
        value 0;
      }
      enum down {
        value 1;
      }
      enum init {
        value 2;
      }
      enum up {
        value 3;
      }
    }
  }
```

```
  typedef multiplier {
    type uint8 {
      range 1..255;
    }
  }

  typedef TTL {
    type uint8 {
      range 1..255;
    }
  }

  typedef bfd-auth-algorithm {
    description "Authentication algorithm";
    type enumeration {
      enum simple-password {
        description
          "Simple password";
      }

      enum keyed-md5 {
        description
          "Keyed message Digest 5";
      }

      enum meticulous-keyed-md5 {
        description
          "Meticulous keyed message Digest 5";
      }

      enum keyed-sha-1 {
        description
          "Keyed secure hash algorithm (SHA1) ";
      }

      enum meticulous-keyed-sha-1 {
        description
          "Meticulous keyed secure hash algorithm (SHA1) ";
      }
    }
  }

  typedef vrfName {
    description "VRF Name";
    type string;
  }

  feature bfd-centralized-session-config {
```

```
    description "BFD session centralized config supported";
  }
  feature bfd-interface-config {
    description "BFD per-interface config supported";
  }
  feature bfd-authentication {
    description "BFD authentication supported";
  }

  grouping bfd-grouping-common-cfg-parms {
    description "BFD grouping for common config parameters";

    leaf local-multiplier {
      description "Local multiplier";
      type multiplier;
      default 3;
    }

    leaf desired-min-tx-interval {
      description
        "Desired minimum transmit interval of control packets";
      type uint32;
      units microseconds;
      mandatory true;
    }

    leaf required-min-rx-interval {
      description
        "Required minimum receive interval of control packets";
      type uint32;
      units microseconds;
      mandatory true;
    }

    leaf demand-enabled {
      description "To enbale demand mode";
      type boolean;
      default false;
    }

    leaf enable-authentication {
      description
        "If set, the Authentication Section is present and the session
         is to be authenticated (see RFC5880 section 6.7 for details).";
      type boolean;
      default false;
    }
```

```
    container authentication-parms {
      if-feature bfd-authentication;
      leaf key-chain-name {
        description
          "Key chain name";
        must "../algorithm" {
          error-message
            "May not be configured without algorithm";
        }
        type string;
      }
      leaf algorithm {
        description "Authentication algorithm to be used";
        must "../key-chain" {
          error-message
            "May not be configured without key-chain";
        }
        type bfd-auth-algorithm;
      }
    }
  }

  grouping bfd-grouping-echo-cfg-parms {
    description "BFD grouping for echo config parameters";
    leaf desired-min-echo-tx-interval {
      description "Desired minumum transmit interval for echo";
      type uint32;
      units microseconds;
      default 0;
    }

    leaf required-min-echo-rx-interval {
      description "Required minimum receive interval for echo";
      type uint32;
      units microseconds;
      default 0;
    }
  }

  grouping bfd-client-base-cfg-parms {
    description
      "BFD grouping for base config parameters which could be used
       by a protocol which is a client of BFD";

    container bfd-cfg {
      leaf enabled {
        type boolean;
        description "True if BFD is enabled";
```

```
        default false;
      }

      leaf local-multiplier {
        type multiplier;
        default 3;
      }

      leaf desired-min-tx-interval {
        description
          "Desired minimum transmit interval of control packets";
        type uint32;
        units microseconds;
        mandatory true;
      }

      leaf required-min-rx-interval {
        description
          "Required minimum receive interval of control packets";
        type uint32;
        units microseconds;
        mandatory true;
      }
    }
  }

  grouping bfd-client-full-cfg-parms {
    description
      "BFD grouping for complete config parameters which could be used
       by a protocol which is a client of BFD.";

    container bfd-cfg {
      leaf enabled {
        type boolean;
        description "True if BFD is enabled";
        default false;
      }

      uses bfd-grouping-common-cfg-parms;

      uses bfd-grouping-echo-cfg-parms;
    }
  }

  grouping bfd-all-session {
    description "BFD session operational information";
    leaf sesssion-type {
      description
```

```
        "BFD session type, this indicates the path type that BFD is
        running on";
      type enumeration {
        enum ip-single-hop {
          value "0";
          description "IP single hop";
        }
        enum ip-multi-hop {
          value "1";
          description "IP multi hop";
        }
      }
    }
    leaf local-discriminator {
      description "Local discriminator";
      type discriminator;
    }
    leaf remote-discriminator {
      description "Remote discriminator";
      type discriminator;
    }
    leaf remote-multiplier {
      description "Remote multiplier";
      type multiplier;
    }
    leaf out-interface {
      description "Outgoing physical interface name";
      type if:interface-ref;
    }
    leaf demand-capability{
      description "Local demand mode capability";
      type boolean;
    }

    list session-running {
      description "BFD session running information";
      leaf local-state {
        type state;
      }
      leaf remote-state {
        type state;
      }
      leaf local-diagnostic {
        type diagnostic;
      }
      leaf remote-diagnostic {
        type diagnostic;
      }
```

```
      leaf detect-Mode {
        description "Detect mode";
        type enumeration {
          enum async-with-echo {
            value "0";
            description "Async with echo";
          }
          enum async-without-echo {
            value "1";
            description "Async without echo";
          }
          enum demand-with-echo {
            value "2";
            description "Demand with echo";
          }
          enum demand-without-echo {
            value "3";
            description "Demand without echo";
          }
        }
      }
      leaf actual-tx-interval {
        description "Actual transmit interval";
        type uint32;
        units microseconds;
      }
      leaf actual-rx-interval {
        description "Actual receive interval";
        type uint32;
        units microseconds;
      }
      leaf actual-echo-tx-interval {
        description "Actual echo transmit interval";
        type uint32;
        units microseconds;
      }
      leaf detect-time {
        description "Detect time";
        type uint32;
        units microseconds;
      }
    }

    list sesssion-statistics {
      description "BFD session statistics";

      leaf create-time {
        description
```

```
          "Time and date when session was created";
        type yang:date-and-time;
      }
      leaf last-down-time {
        description
          "Time and date of last time the session went down";
        type yang:date-and-time;
      }
      leaf last-up-time {
        description
          "Time and date of last time the session went up";
        type yang:date-and-time;
      }
      leaf receive-pkt {
        description "Received Packet Count";
        type uint64;
      }
      leaf send-pkt {
        description "Sent Packet Count";
        type uint64;
      }
      leaf down-count {
        description "Session Down Count";
        type uint32;
      }
      leaf receive-bad-pkt {
        description "Received bad packet count";
        type uint64;
      }
      leaf send-failed-pkt {
        description "Packet Failed to Send Count";
        type uint64;
      }
      leaf short-break-count {
        description "Shortbreak count";
        default "0";
        type uint32;
      }
    }
  }

  container bfd-cfg {
    container bfd-session-cfg {
      if-feature bfd-centralized-session-config;
      list session-ip-sh {
        key "interface dest-addr";
        leaf interface {
          description
```

```
            "Interface on which the IP single-hop session is running.";
          type if:interface-ref;
        }
        leaf dest-addr {
          description
            "IP address of the peer";
          type inet:ip-address;
        }
        leaf admin-down {
          description
            "Is the BFD session administratively down";
          type boolean;
          default false;
        }
        uses bfd-grouping-common-cfg-parms;

        uses bfd-grouping-echo-cfg-parms;
      }

      list session-ip-mh {

        key "vrf-name source-addr dest-addr";
        leaf vrf-name {
          description "Routing instance";
          type vrfName;
        }
        leaf source-addr {
          description
            "Local IP address";
          type inet:ip-address;
        }
        leaf dest-addr {
          description
            "IP address of the peer";
          type inet:ip-address;
        }
        leaf admin-down {
          description
            "Is the BFD session administratively down";
          type boolean;
          default false;
        }
        uses bfd-grouping-common-cfg-parms;

        leaf tx-ttl {
          type TTL;
          default 255;
          description "TTL of outgoing BFD control packets";
```

```
        }
        leaf rx-ttl {
          type TTL;
          description
            "Minimum allowed TTL value for incoming BFD control
            packets";
          mandatory true;
        }
      }
    }

    list bfd-interface-cfg {
      if-feature bfd-interface-config;

      description "Per-interface BFD configuration";
      key interface;
      leaf interface {
        type if:interface-ref;
      }
      uses bfd-grouping-common-cfg-parms;

      uses bfd-grouping-echo-cfg-parms;

    }
  }

  container bfd-oper {
    container bfd-session-statistics {
      config "false";
      description "BFD session number";
      leaf ip-sh-session-num {
        description "IP single hop session number";
        config "false";
        type uint32;
      }
      leaf ip-mh-session-num {
        description "IP multi hop session Number";
        config "false";
        type uint32;
      }
      leaf total-session-num {
        description "Total session number";
        config "false";
        type uint32;
      }
      leaf session-up-num {
        description "Session up number";
        config "false";
```

```
        type uint32;
      }
      leaf sess-down-num {
        description "Session down number";
        config "false";
        type uint32;
      }
    }

    container bfd-session-lists {
      config false;
      list session-ip-sh {
        key "interface dest-addr";
        leaf interface {
          type if:interface-ref;
        }
        leaf dest-addr {
          type inet:ip-address;
        }

        uses bfd-all-session;
      }

      list session-ip-mh {
        key "vrfName source-addr dest-addr";
        leaf vrfName {
          type vrfName;
        }
        leaf source-addr {
          type inet:ip-address;
        }
        leaf dest-addr {
          type inet:ip-address;
        }
        leaf ttl {
          description "TTL of session";
          config "false";
          type TTL;
        }
        uses bfd-all-session;
      }
    }
  }

  grouping bfd-notification-params {
    description
      "This group describes common params that will be send
        as part of BFD notification";
```

```
    leaf local-discr {
      description "BFD local discriminator";
      type discriminator;
    }

    leaf remote-discr {
      description "BFD remote discriminator";
      type discriminator;
    }

    leaf new-state {
      description "Current BFD state";
      type state;
    }

    leaf state-change-reason {
      description "BFD state change reason";
      type string;
    }

    leaf time-in-previous-state {
      description "How long the BFD session was in the previous state";
      type string;
    }

    leaf dest-addr {
      description "BFD peer address";
      type inet:ip-address;
    }
  }

  notification bfd-singlehop-notification {

    uses bfd-notification-params;

    leaf interface {
      description "Interface to which this BFD session belongs to";
      type if:interface-ref;
    }

    leaf echo-enabled {
      description "Was echo enabled for BFD";
      type boolean;
    }
  }


  notification bfd-multihop-notification {
```

```
    uses bfd-notification-params;

    leaf vrf-name {
      description "Routing instance";
      type vrfName;
    }

    leaf source-addr {
      description "BFD local address";
      type inet:ip-address;
    }
  }
}
```

2.11. BFD Client Example Configuration Yang Module

```
   module bfd-routing-app {
      namespace "urn:cisco:params:xml:ns:yang:bfdroutingapp";
      prefix bfd-routing-app;

      import bfd {
        prefix "bfd";
      }

      import ietf-interfaces {
        prefix "if";
      }

      organization
        "ACME";
      contact
        "acme@acme.com";

      description
        "Testing BFD grouping (simulating a routing application)";

      revision 2015-02-14 {
        description
          "Initial revision.";
      }

      feature routing-app-bfd {
        description "BFD configuration under routing-app";
      }

      list area {
```

```
        description
          "Specify a routing area.";

        key "area-id";

        leaf area-id {
          type uint32;
        }

        uses bfd:bfd-client-base-cfg-parms {
          if-feature routing-app-bfd;
        }

        list interface {
          key "interface";
          leaf interface {
            type if:interface-ref;
          }
          uses bfd:bfd-client-base-cfg-parms {
            if-feature routing-app-bfd;
          }
        }

      }
   }
```

2.12. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory to implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

The YANG module has writeable data nodes which can be used for creation of BFD sessions and modification of BFD session parameters. The system should "police" creation of BFD sessions to prevent new sessions from causing existing BFD sessions to fail. For BFD session modification, the BFD protocol has mechanisms in place which allow for in service modification.

2.13. IANA Considerations

The IANA is requested to as assign a new new namespace URI from the IETF XML registry.

URI:TBD

2.14. Acknowledgements

We would also like to thank Nobo Akiya and Jeff Haas for their encouragement on this work.

3. References

3.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, June 2010.

[RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, June 2010.

[RFC5882] Katz, D. and D. Ward, "Generic Application of Bidirectional Forwarding Detection (BFD)", RFC 5882, June 2010.

[RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths", RFC 5883, June 2010.

[RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow, "Bidirectional Forwarding Detection (BFD) for MPLS Label Switched Paths (LSPs)", RFC 5884, June 2010.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC7130] Bhatia, M., Chen, M., Boutros, S., Binderberger, M., and J. Haas, "Bidirectional Forwarding Detection (BFD) on Link Aggregation Group (LAG) Interfaces", RFC 7130, February 2014.

3.2. Informative References

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-04 (work in progress), January 2015.

Authors' Addresses

Lianshu Zheng (editor)
Huawei Technologies
China

Email: vero.zheng@huawei.com

Reshad Rahman (editor)
Cisco Systems
USA

Email: rrahman@cisco.com

Santosh Pallagatti
Juniper Networks
India

Email: santoshpk@juniper.net

Mahesh Jethanandani
Ciena Corporation

Email: mjethanandani@gmail.com