Routing Working Group   A. Mishra
Internet-Draft   Ciena Corporation
Intended status: Standards Track   M. Jethanandani
Expires: July 30, 2017   Cisco Systems
                                                             A. Saxena
                                                     Ciena Corporation
                                                          S. Pallagatti
                                                      Juniper Networks
                                                               M. Chen
                                                                Huawei
                                                                P. Fan
                                                          China Mobile
                                                      January 26, 2017

                               BFD Stability
                   draft-ashesh-bfd-stability-05.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 RFC 2119 [RFC2119].

   Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   The Bidirectional Forwarding Detection (BFD) [RFC5880] 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 in the Detection Time, as
   defined in BFD [RFC5880] 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 to measure the
   stability of BFD sessions and provides data to the operators for the
   cause of a BFD failure.

   This document does not propose BFD extension to measure data traffic
   loss or delay on a link or tunnel and the scope is limited to BFD
   frames.

2. Use Cases

   Legacy BFD cannot detect any BFD frame loss if loss does not last for
   dead interval. This draft proposes a method to detect a dropped
   frame on the receiver. For example, if the receiver receives BFD CC
   frame k at time t but receives frame k+3 at time t+10ms, and never
   receives frame k+1 and/or k+2, then it has experienced a drop.

   This proposal enables BFD engine to generate diagnostic information
   on the health of each BFD session that could be used to preempt a
   failure on a link that BFD was monitoring by allowing time for a
   corrective action to be taken.

   In a faulty datapath scenario, operator can use BFD health
   information to trigger delay and loss measurement OAM protocol
   (Connectivity Fault Management (CFM) or Loss Measurement (LM)-Delay
   Measurement (DM)) to further isolate the issue.

3. BFD Null-Authentication TLV

   The functionality proposed for BFD stability measurement is achieved
   by appending the Null-Authentication TLV (as defined in Optimizing
   BFD Authentication [I-D.ietf-bfd-optimizing-authentication]) to the
   BFD control frame that do not have authentication enabled.

4. Theory of Operations

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

   When using MD5 or SHA authentication, BFD uses authentication TLV
   that carries the Sequence Number. However, if non-meticulous
   authentication is being used, or no authentication is in use, then
   the non-authenticated BFD frames MUST include NULL-Auth TLV.

4.1. Loss Measurement

   Loss measurement counts the number of BFD control frames missed at
   the receiver during any Detection Time period. The loss is detected
   by comparing the Sequence Number field in the Auth TLV (NULL or
   otherwise) 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 NULL-Auth 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.

When the Sequence Number wraps around it should start from 1 instead
of 0.

5. IANA Requirements

   N/A

6. Security Consideration

   Other than concerns raised in BFD [RFC5880] there are no new concerns
   with this proposal.

7. Contributors

   Manav Bhatia

8. Acknowledgements

   Authors would like to thank Nobo Akiya, Jeffery Haas, Peng Fan,
   Dileep Singh, Basil Saji, Sagar Soni and Mallik Mudigonda who also
   contributed to this document.

9. Normative References

   [I-D.ietf-bfd-optimizing-authentication]
             Jethanandani, M., Mishra, A., Saxena, A., and M. Bhatia,
             "Optimizing BFD Authentication", draft-ietf-bfd-
             optimizing-authentication-02 (work in progress), January
             2017.

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

   [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection
             (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010,
             <http://www.rfc-editor.org/info/rfc5880>.

Authors' Addresses

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

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

Mahesh Jethanandani
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
USA

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

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

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

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

Email: santoshpk@juniper.net

Mach Chen
Huawei

Email: mach.chen@huawei.com

      Peng Fan
      China Mobile
      32 Xuanwumen West Street
      Beijing, Beijing
      China

      Email: fanp08@gmail.com

Internet Engineering Task Force   D. Katz
Internet-Draft   Juniper Networks
Updates: 5880 (if approved)   D. Ward
Intended status: Standards Track   Cisco Systems
Expires: June 16, 2019   S. Pallagatti, Ed.
                                                                Rtbrick
                                                          G. Mirsky, Ed.
                                                               ZTE Corp.
                                                       December 13, 2018

                       BFD for Multipoint Networks
                      draft-ietf-bfd-multipoint-19

Abstract

   This document describes extensions to the Bidirectional Forwarding
   Detection (BFD) protocol for its use in multipoint and multicast
   networks.

   This document updates RFC 5880.

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 https://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 June 16, 2019.

Copyright Notice

Table of Contents

1. Introduction

   The Bidirectional Forwarding Detection protocol [RFC5880] specifies a
   method for verifying unicast connectivity between a pair of systems.
   This document updates [RFC5880] by defining a new method for using

BFD. This new method provides verification of multipoint or
multicast connectivity between a multipoint sender (the "head") and a
set of one or more multipoint receivers (the "tails").

As multipoint transmissions are inherently unidirectional, this
mechanism purports only to verify this unidirectional connectivity.
Although this seems in conflict with the "Bidirectional" in BFD, the
protocol is capable of supporting this use case. Use of BFD in
Demand mode allows a tail to monitor the availability of a multipoint
path even without the existence of some kind of a return path to the
head. As an option, if a return path from a tail to the head exists,
the tail may notify the head of the lack of multipoint connectivity.
Details of tail notification to the head are outside the scope of
this document and are discussed in
[I-D.ietf-bfd-multipoint-active-tail].

This application of BFD allows for the tails to detect a lack of
connectivity from the head. For some applications such detection of
the failure at the tail is useful. For example, use of multipoint
BFD to enable fast failure detection and faster failover in multicast
VPN described in [I-D.ietf-bess-mvpn-fast-failover]. Due to
unidirectional nature, virtually all options and timing parameters
are controlled by the head.

Throughout this document, the term "multipoint" is defined as a
mechanism by which one or more systems receive packets sent by a
single sender. This specifically includes such things as IP
multicast and point-to-multipoint MPLS.

The term "connectivity" in this document is not being used in the
context of connectivity verification in transport network but as an
alternative to "continuity", i.e., the existence of a forwarding path
between the sender and the receiver.

This document effectively updates and extends the base BFD
specification [RFC5880].

2. Keywords

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

3. Goals

   The primary goal of this mechanism is to allow tails to rapidly
   detect the fact that multipoint connectivity from the head has
   failed.

   Another goal is for the mechanism to work on any multicast
   technology.

   A further goal is to support multiple, overlapping point-to-
   multipoint paths, as well as multipoint-to-multipoint paths, and to
   allow point-to-point BFD sessions to operate simultaneously among the
   systems participating in Multipoint BFD.

   It is not a goal for this protocol to verify point-to-point bi-
   directional connectivity between the head and any tail. This can be
   done independently (and with no penalty in protocol overhead) by
   using point-to-point BFD.

4. Overview

   The heart of this protocol is the periodic transmission of BFD
   Control packets along a multipoint path, from the head to all tails
   on the path. The contents of the BFD packets provide the means for
   the tails to calculate the detection time for path failure. If no
   BFD Control packets are received by a tail for a detection time, the
   tail declares that the path has failed. For some applications this
   is the only mechanism necessary; the head can remain ignorant of the
   status of connectivity to the tails.

   The head of a multipoint BFD session may wish to be alerted to the
   tails' connectivity (or lack thereof). Details of how the head keeps
   track of tails and how tails alert their connectivity to the head are
   outside the scope of this document and are discussed in
   [I-D.ietf-bfd-multipoint-active-tail].

   Although this document describes a single head and a set of tails
   spanned by a single multipoint path, the protocol is capable of
   supporting (and discriminating between) more than one multipoint path
   at both heads and tails, as described in Section 5.7 and
   Section 5.13.2. Furthermore, the same head and tail may share
   multiple multipoint paths, and a multipoint path may have multiple
   heads.

5. Protocol Details

   This section describes the operation of Multipoint BFD in detail.

5.1. Multipoint BFD Control Packets

   Multipoint BFD Control packets (packets sent by the head over a
   multipoint path) are explicitly marked as such, via the setting of
   the M bit [RFC5880]. This means that Multipoint BFD does not depend
   on the recipient of a packet to know whether the packet was received
   over a multipoint path. This can be useful in scenarios where this
   information may not be available to the recipient.

5.2. Session Model

   Multipoint BFD is modeled as a set of sessions of different types.
   The elements of procedure differ slightly for each type.

   The head has a session of type MultipointHead, as defined in
   Section 5.4.1, 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 has a session of type MultipointTail, as defined in
   Section 5.4.1, associated with a multipoint path. These sessions
   receive BFD Control packets from the head over the multipoint path.

5.3. Session Failure Semantics

   The semantics of session failure is subtle enough to warrant further
   explanation.

   MultipointHead sessions cannot fail (since they are controlled
   administratively).

   If a MultipointTail session fails, it means that the tail definitely
   has lost contact with the head (or the head has been administratively
   disabled) and the tail may use mechanisms other than BFD, e.g.,
   logging or NETCONF [RFC6241], to send a notification to the user.

5.4. State Variables

   Multipoint BFD introduces some new state variables and modifies the
   usage of a few existing ones.

5.4.1. New State Variable Values

   A number of new values of the state variable bfd.SessionType are
   added to the base BFD [RFC5880] and base S-BFD [RFC7880]
   specifications in support of Multipoint BFD.

      bfd.SessionType

         The type of this session as defined in [RFC7880]. Newly added
         values are:

            PointToPoint: Classic point-to-point BFD, as described in
            [RFC5880].

            MultipointHead: A session on the head responsible for the
            periodic transmission of multipoint BFD Control packets
            along the multipoint path.

            MultipointTail: A multipoint session on a tail.

         This variable MUST be initialized to the appropriate type when
         the session is created.

5.4.2. State Variable Initialization and Maintenance

   Some state variables defined in section 6.8.1 of [RFC5880] need to be
   initialized or manipulated differently depending on the session type.

      bfd.RequiredMinRxInterval

         This variable MUST be initialized to 0 for session type
         MultipointHead.

      bfd.DemandMode

         This variable MUST be initialized to 1 for session type
         MultipointHead and MUST be initialized to 0 for session type
         MultipointTail.

5.5. State Machine

   The BFD state machine works slightly differently in the multipoint
   application. In particular, since there is a many-to-one mapping,
   three-way handshakes for session establishment and teardown are
   neither possible nor appropriate. As such, there is no Init state.
   Sessions of type MultipointHead MUST NOT send BFD control packets
   with the State field being set to INIT, and those packets MUST be
   ignored on receipt.

The following diagram provides an overview of the state machine for
session type MultipointTail. The notation on each arc represents the
state of the remote system (as received in the State field in the BFD
Control packet) or indicates the expiration of the Detection Timer.

```
                           DOWN, ADMIN DOWN,
                  +------+  TIMER                   +------+
            +----|      |<--------------------|      |----+
   DOWN,    |    | DOWN |                     |  UP  |    |UP
ADMIN DOWN,+--->|      |-------------------->|      |<---+
   TIMER        +------+        UP            +------+
```
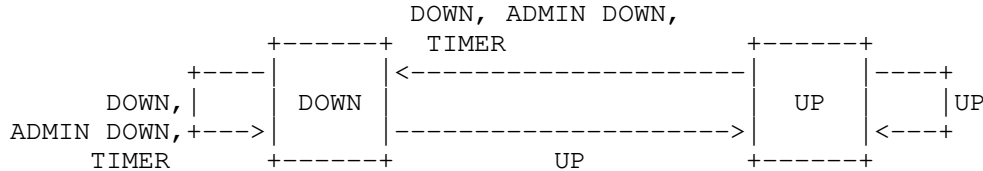
Sessions of type MultipointHead never receive packets and have no
Detection Timer, and as such all state transitions are
administratively driven.

5.6. Session Establishment

   Unlike point-to-point BFD, Multipoint BFD provides a form of the
   discovery mechanism for tails to discover the head. The minimum
   amount of a priori information required both on the head and tails is
   the binding to the multipoint path over which BFD is running. The
   head transmits Multipoint BFD packets on that path, and the tails
   listen for BFD packets on that path. All other information can be
   determined dynamically.

   A session of type MultipointHead is created for each multipoint path
   over which the head wishes to run BFD. This session runs in the
   Active role, per section 6.1 [RFC5880]. Except when administratively
   terminating BFD service, this session is always in state Up and
   always operates in Demand mode. No received packets are ever
   demultiplexed to the MultipointHead session. In this sense, it is a
   degenerate form of a session.

   Sessions on the tail MAY be established dynamically, based on the
   receipt of a Multipoint BFD Control packet from the head, and are of
   type MultipointTail. Tail sessions always take the Passive role, per
   section 6.1 [RFC5880].

5.7. Discriminators and Packet Demultiplexing

   The use of Discriminators is somewhat different in Multipoint BFD
   than in Point-to-point BFD.

   The head sends Multipoint BFD Control packets over the multipoint
   path via the MultipointHead session with My Discriminator set to a

value bound to the multipoint path, and with Your Discriminator set
to zero.

IP and MPLS multipoint tails MUST demultiplex BFD packets based on a
combination of the source address, My Discriminator and the identity
of the multipoint path which the Multipoint BFD Control packet was
received from. Together they uniquely identify the head of the
multipoint path. Bootstrapping a BFD session to multipoint MPLS LSP
may use the control plane, e.g., as described in
[I-D.ietf-bess-mvpn-fast-failover], and is outside the scope of this
document.

Note that, unlike point-to-point sessions, the My Discriminator value
on MultipointHead session MUST NOT be changed during the life of a
session. This is a side effect of the more complex demultiplexing
scheme.

5.8. Packet consumption on tails

BFD packets received on tails for an IP multicast group MUST be
consumed by tails and MUST NOT be forwarded to receivers. Nodes with
the BFD session of type MultipointTail MUST identify packets received
on an IP multipoint path as BFD control packet if the destination UDP
port value equals 3784.

For multipoint LSPs, when IP/UDP encapsulation of BFD control packets
is used, MultipointTail MUST expect destination UDP port 3784.
Destination IP address of BFD control packet MUST be in 127.0.0.0/8
range for IPv4 or in 0:0:0:0:0:FFFF:7F00:0/104 range for IPv6. The
use of these destination addresses is consistent with the
explanations and usage in [RFC8029]. Packets identified as BFD
packets MUST be consumed by MultipointTail and demultiplexed as
described in Section 5.13.2. Use of other types of encapsulation of
the BFD control message over multipoint LSP is outside the scope of
this document.

5.9. Bringing Up and Shutting Down Multipoint BFD Service

Because there is no three-way handshake in Multipoint BFD, a newly
started head (that does not have any previous state information
available) SHOULD start with bfd.SessionState set to Down and
bfd.RequiredMinRxInterval MUST be set to zero in the MultipointHead
session. The session SHOULD remain in this state for a time equal to
(bfd.DesiredMinTxInterval * bfd.DetectMult). This will ensure that
all MultipointTail sessions are reset (so long as the restarted head
is using the same or a larger value of bfd.DesiredMinTxInterval than
it did previously).

Multipoint BFD service is brought up by administratively setting
bfd.SessionState to Up in the MultipointHead session.

The head of a multipoint BFD session may wish to shut down its BFD
service in a controlled fashion. This is desirable because the tails
need not wait a detection time prior to declaring the multipoint
session to be down (and taking whatever action is necessary in that
case).

To shut down a multipoint session in a controlled fashion the head
MUST administratively set bfd.SessionState in the MultipointHead
session to either Down or AdminDown and SHOULD set
bfd.RequiredMinRxInterval to zero. The session SHOULD send BFD
Control packets in this state for a period equal to
(bfd.DesiredMinTxInterval * bfd.DetectMult). Alternatively, the head
MAY stop transmitting BFD Control packets and not send any more BFD
Control packets with the new state (Down or AdminDown). Tails will
declare the multipoint session down only after the detection time
interval runs out.

5.10. Timer Manipulation

Because of the one-to-many mapping, a session of type MultipointHead
SHOULD NOT initiate a Poll Sequence in conjunction with timer value
changes. However, to indicate a change in the packets,
MultipointHead session MUST send packets with the P bit set.
MultipointTail session MUST NOT reply if the packet has M and P bits
set and bfd.RequiredMinRxInterval set to 0. Because the Poll
Sequence is not used, the tail cannot negotiate down MultpointHead's
transmit interval. If the value of Desired Min TX Interval in the
BFD Control packet received by MultipointTail is too high (that
determination may change in time based on the current environment) it
must be handled by the implementation and may be controlled by local
policy, e.g., close the MultipointTail session.

The MultipointHead, when changing the transmit interval to a higher
value, MUST send BFD control packets with P bit set at the old
transmit interval before using the higher value in order to avoid
false detection timeouts at the tails. MultipointHead session MAY
also wait some amount of time before making the changes to the
transmit interval (through configuration).

Change in the value of bfd.RequiredMinRxInterval is outside the scope
of this document and is discussed in
[I-D.ietf-bfd-multipoint-active-tail].

5.11. Detection Times

   Multipoint BFD is inherently asymmetric. As such, each session type
   has a different approach to detection times.

   Since MultipointHead sessions never receive packets, they do not
   calculate a detection time.

   MultipointTail sessions cannot influence the transmission rate of the
   MultipointHead session using the Required Min Rx Interval field
   because of its one-to-many nature. As such, the detection time
   calculation for a MultipointTail session does not use
   bfd.RequiredMinRxInterval. The detection time is calculated as the
   product of the last received values of Desired Min TX Interval and
   Detect Mult.

   The value of bfd.DetectMult may be changed at any time on any session
   type.

5.12. State Maintenance for Down/AdminDown Sessions

   The length of time session state is kept after the session goes down
   determines how long the session will continue to send BFD Control
   packets (since no packets can be sent after the session is
   destroyed).

5.12.1. MultipointHead Sessions

   When a MultipointHead session transitions to states Down or
   AdminDown, the state SHOULD be maintained for a period equal to
   (bfd.DesiredMinTxInterval * bfd.DetectMult) to ensure that the tails
   more quickly detect the session going down (by continuing to transmit
   BFD Control packets with the new state).

5.12.2. MultipointTail Sessions

   MultipointTail sessions MAY be destroyed immediately upon leaving Up
   state, since tail will transmit no packets.

   Otherwise, MultipointTail sessions SHOULD be maintained as long as
   BFD Control packets are being received by it (which by definition
   will indicate that the head is not Up).

5.13. Base Specification Text Replacement

   The following sections are meant to replace the corresponding
   sections in the base specification [RFC5880] in support of BFD for

   multipoint networks while not changing processing for point-to-point
   BFD.

5.13.1. Reception of BFD Control Packets

   The following procedure replaces the entire section 6.8.6 of
   [RFC5880].

   When a BFD Control packet is received, the following procedure MUST
   be followed, in the order specified. If the packet is discarded
   according to these rules, processing of the packet MUST cease at that
   point.

      If the version number is not correct (1), the packet MUST be
      discarded.

      If the Length field is less than the minimum correct value (24 if
      the A bit is clear, or 26 if the A bit is set), the packet MUST be
      discarded.

      If the Length field is greater than the payload of the
      encapsulating protocol, the packet MUST be discarded.

      If the Detect Mult field is zero, the packet MUST be discarded.

      If the My Discriminator field is zero, the packet MUST be
      discarded.

      Demultiplex the packet to a session according to Section 5.13.2
      below. The result is either a session of the proper type, or the
      packet is discarded (and packet processing MUST cease).

      If the A bit is set and no authentication is in use (bfd.AuthType
      is zero), the packet MUST be discarded.

      If the A bit is clear and authentication is in use (bfd.AuthType
      is nonzero), the packet MUST be discarded.

      If the A bit is set, the packet MUST be authenticated under the
      rules of [RFC5880] section 6.7, based on the authentication type
      in use (bfd.AuthType). This may cause the packet to be discarded.

      Set bfd.RemoteDiscr to the value of My Discriminator.

      Set bfd.RemoteState to the value of the State (Sta) field.

      Set bfd.RemoteDemandMode to the value of the Demand (D) bit.

Set bfd.RemoteMinRxInterval to the value of Required Min RX
Interval.

If the Required Min Echo RX Interval field is zero, the
transmission of Echo packets, if any, MUST cease.

If a Poll Sequence is being transmitted by the local system and
the Final (F) bit in the received packet is set, the Poll Sequence
MUST be terminated.

If bfd.SessionType is PointToPoint, update the transmit interval
as described in [RFC5880] section 6.8.2.

If bfd.SessionType is PointToPoint, update the Detection Time as
described in section 6.8.4 of [RFC5880].

Else

    If bfd.SessionType is MultipointTail, then update the Detection
    Time as the product of the last received values of Desired Min
    TX Interval and Detect Mult, as described in Section 5.11 of
    this specification.

If bfd.SessionState is AdminDown

    Discard the packet

If the received state is AdminDown

    If bfd.SessionState is not Down

        Set bfd.LocalDiag to 3 (Neighbor signaled session down)

        Set bfd.SessionState to Down

Else

    If bfd.SessionState is Down

        If bfd.SessionType is PointToPoint

            If received State is Down

                Set bfd.SessionState to Init

            Else if received State is Init

                Set bfd.SessionState to Up

        Else (bfd.SessionType is not PointToPoint)

            If received State is Up

                Set bfd.SessionState to Up

        Else if bfd.SessionState is Init

            If received State is Init or Up

                Set bfd.SessionState to Up

        Else (bfd.SessionState is Up)

            If received State is Down

                Set bfd.LocalDiag to 3 (Neighbor signaled session down)

                Set bfd.SessionState to Down

   Check to see if Demand mode should become active or not (see
   [RFC5880] section 6.6).

   If bfd.RemoteDemandMode is 1, bfd.SessionState is Up and
   bfd.RemoteSessionState is Up, Demand mode is active on the remote
   system and the local system MUST cease the periodic transmission
   of BFD Control packets (see Section 5.13.3).

   If bfd.RemoteDemandMode is 0, or bfd.SessionState is not Up, or
   bfd.RemoteSessionState is not Up, Demand mode is not active on the
   remote system and the local system MUST send periodic BFD Control
   packets (see Section 5.13.3).

   If the Poll (P) bit is set, and bfd.SessionType is PointToPoint,
   send a BFD Control packet to the remote system with the Poll (P)
   bit clear, and the Final (F) bit set (see Section 5.13.3).

   If the packet was not discarded, it has been received for purposes
   of the Detection Time expiration rules in [RFC5880] section 6.8.4.

5.13.2. Demultiplexing BFD Control Packets

   This section is part of the replacement for [RFC5880] section 6.8.6,
   separated for clarity.

      If the Multipoint (M) bit is set

If the Your Discriminator field is nonzero, the packet MUST be discarded.

Select a session as based on source address, My Discriminator and the identity of the multipoint path which the Multipoint BFD Control packet was received.

If a session is found, and bfd.SessionType is not MultipointTail, the packet MUST be discarded.

Else

If a session is not found, a new session of type MultipointTail MAY be created, or the packet MAY be discarded. This choice can be controlled by the local policy, e.g., by settinga maximum number of MultipointTail sessions. Use of the local policy and the exact mechanism of it are outside the scope of this specification.

Else (Multipoint bit is clear)

If the Your Discriminator field is nonzero

Select a session based on the value of Your Discriminator. If no session is found, the packet MUST be discarded.

Else (Your Discriminator is zero)

If the State field is not Down or AdminDown, the packet MUST be discarded.

Otherwise, the session MUST be selected based on some combination of other fields, possibly including source addressing information, the My Discriminator field, and the interface over which the packet was received. The exact method of selection is application-specific and is thus outside the scope of this specification.

If a matching session is found, and bfd.SessionType is not PointToPoint, the packet MUST be discarded.

If a matching session is not found, a new session of type PointToPoint MAY be created, or the packet MAY be discarded. This choice MAY be controlled by a local policy and is outside the scope of this specification.

If the State field is Init and bfd.SessionType is not PointToPoint, the packet MUST be discarded.

5.13.3. Transmitting BFD Control Packets

   The following procedure replaces the entire section 6.8.7 of
   [RFC5880].

   With the exceptions listed in the remainder of this section, a system
   MUST NOT transmit BFD Control packets at an interval less than the
   larger of bfd.DesiredMinTxInterval and bfd.RemoteMinRxInterval, less
   applied jitter (see below). In other words, the system reporting the
   slower rate determines the transmission rate.

   The periodic transmission of BFD Control packets MUST be jittered on
   a per-packet basis by up to 25%, that is, the interval MUST be
   reduced by a random value of 0 to 25%, in order to avoid self-
   synchronization with other systems on the same subnetwork. Thus, the
   average interval between packets will be roughly 12.5% less than that
   negotiated.

   If bfd.DetectMult is equal to 1, the interval between transmitted BFD
   Control packets MUST be no more than 90% of the negotiated
   transmission interval, and MUST be no less than 75% of the negotiated
   transmission interval. This is to ensure that, on the remote system,
   the calculated Detection Time does not pass prior to the receipt of
   the next BFD Control packet.

   A system MUST NOT transmit any BFD Control packets if bfd.RemoteDiscr
   is zero and the system is taking the Passive role.

   A system MUST NOT transmit any BFD Control packets if bfd.SessionType
   is MultipointTail.

   A system MUST NOT periodically transmit BFD Control packets if Demand
   mode is active on the remote system (bfd.RemoteDemandMode is 1,
   bfd.SessionState is Up, and bfd.RemoteSessionState is Up) and a Poll
   Sequence is not being transmitted.

   A system MUST NOT periodically transmit BFD Control packets if
   bfd.RemoteMinRxInterval is zero.

   If bfd.SessionType is MultipointHead, the transmit interval MUST be
   set to bfd.DesiredMinTxInterval (this should happen automatically, as
   bfd.RemoteMinRxInterval will be zero).

   If bfd.SessionType is not MultipointHead, the transmit interval MUST
   be recalculated whenever bfd.DesiredMinTxInterval changes, or
   whenever bfd.RemoteMinRxInterval changes, and is equal to the greater
   of those two values. See [RFC5880] sections 6.8.2 and 6.8.3 for
   details on transmit timers.

A system MUST NOT set the Demand (D) bit if bfd.SessionType is
MultipointTail.

A system MUST NOT set the Demand (D) bit if bfd.SessionType
PointToPoint unless bfd.DemandMode is 1, bfd.SessionState is Up, and
bfd.RemoteSessionState is Up.

If bfd.SessionType is PointToPoint or MultipointHead, a BFD Control
packet SHOULD be transmitted during the interval between periodic
Control packet transmissions when the contents of that packet would
differ from that in the previously transmitted packet (other than the
Poll and Final bits) in order to more rapidly communicate a change in
state.

The contents of transmitted BFD Control packets MUST be set as
follows:

   Version

      Set to the current version number (1).

   Diagnostic (Diag)

      Set to bfd.LocalDiag.

   State (Sta)

      Set to the value indicated by bfd.SessionState.

   Poll (P)

      Set to 1 if the local system is sending a Poll Sequence or is a
      session of type MultipointHead soliciting the identities of the
      tails, or 0 if not.

   Final (F)

      Set to 1 if the local system is responding to a Control packet
      received with the Poll (P) bit set, or 0 if not.

   Control Plane Independent (C)

      Set to 1 if the local system's BFD implementation is
      independent of the control plane (it can continue to function
      through a disruption of the control plane).

   Authentication Present (A)

Set to 1 if authentication is in use in this session
(bfd.AuthType is nonzero), or 0 if not.

Demand (D)

Set to bfd.DemandMode if bfd.SessionState is Up and
bfd.RemoteSessionState is Up. Set to 1 if bfd.SessionType is
MultipointHead. Otherwise it is set to 0.

Multipoint (M)

Set to 1 if bfd.SessionType is MultipointHead. Otherwise, it
is set to 0.

Detect Mult

Set to bfd.DetectMult.

Length

Set to the appropriate length, based on the fixed header length
(24) plus any Authentication Section.

My Discriminator

Set to bfd.LocalDiscr.

Your Discriminator

Set to bfd.RemoteDiscr.

Desired Min TX Interval

Set to bfd.DesiredMinTxInterval.

Required Min RX Interval

Set to bfd.RequiredMinRxInterval.

Required Min Echo RX Interval

Set to 0 if bfd.SessionType is MultipointHead or
MultipointTail. Otherwise, set to the minimum required Echo
packet receive interval for this session. If this field is set
to zero, the local system is unwilling or unable to loop back
BFD Echo packets to the remote system, and the remote system
will not send Echo packets.

Authentication Section

Included and set according to the rules in [RFC5880] section
6.7 if authentication is in use (bfd.AuthType is nonzero).
Otherwise, this section is not present.

6. Congestion Considerations

As a foreword, although congestion can occur because of a number of
factors, it should be noted that high transmission rates are by
themselves subject to creating congestion either along the path or at
the tail end(s). As such, as stated in [RFC5883]:

"it is required that the operator correctly provision the rates at
which BFD is transmitted to avoid congestion (e.g link, I/O, CPU)
and false failure detection."

Use of BFD in multipoint networks, as specified in this document,
over multiple hops requires consideration of the mechanisms to react
to network congestion. Requirements stated in Section 7 of the BFD
base specification [RFC5880] equally apply to BFD in multipoint
networks and are repeated here:

"When BFD is used across multiple hops, a congestion control
mechanism MUST be implemented, and when congestion is detected,
the BFD implementation MUST reduce the amount of traffic it
generates."

The mechanism to control the load of BFD traffic MAY use BFD's
configuration interface to control BFD state variable
bfd.DesiredMinTxInterval. However, such a control loop do not form
part of the BFD protocol itself and its specification is thus outside
the scope of this document.

Additional considerations apply to BFD in multipoint networks, as
specified in this document. Indeed, because a tail does not transmit
any BFD Control packets to the head of the BFD session, such head
node has no BFD based mechanism to be aware of the state of the
session at the tail. In the absence of any other mechanism, the head
of the session could thus continue to send packets towards the
tail(s) even though a link failure has happened. In such a scenario
when it is required for the head of the session to be aware of the
state of the tail of the session, it is RECOMMENDED to implement
[I-D.ietf-bfd-multipoint-active-tail].

7. IANA Considerations

   This document has no actions for IANA.

8. Security Considerations

   The same security considerations as those described in [RFC5880]
   apply to this document. Additionally, implementations that create
   MultpointTail sessions dynamically upon receipt of Multipoint BFD
   Control packets MUST implement protective measures to prevent an
   infinite number of MultipointTail sessions being created. Below are
   listed some points to be considered in such implementations.

      If a Multipoint BFD Control packet did not arrive on a multicast
      path (e.g., on the expected interface, with expected MPLS label,
      etc), then a MultipointTail session should not be created.

      If redundant streams are expected for a given multicast stream,
      then the implementations should not create more MultipointTail
      sessions than the number of streams. Additionally, when the
      number of MultipointTail sessions exceeds the number of expected
      streams, then the implementation should generate an alarm to users
      to indicate the anomaly.

      The implementation should have a reasonable upper bound on the
      number of MultipointHead sessions that can be created, with the
      upper bound potentially being computed based on the load these
      would generate.

      The implementation should have a reasonable upper bound on the
      number of MultipointTail sessions that can be created, with the
      upper bound potentially being computed based on the number of
      multicast streams that the system is expecting.

   If authentication is in use, the head and all tails may be configured
   to have a common authentication key in order for the tails to
   validate multipoint BFD Control packets.

   Shared keys in multipoint scenarios allow any tail to spoof the head
   from the viewpoint of any other tail. For this reason, using shared
   keys to authenticate BFD Control packets in multipoint scenarios is a
   significant security exposure unless all tails can be trusted not to
   spoof the head. Otherwise, asymmetric message authentication would
   be needed, e.g., protocols that use Timed Efficient Stream Loss-
   Tolerant Authentication (TESLA) as described in [RFC4082].
   Applicability of the assymetric message authentication to BFD for
   multipoint networks is ouside the scope of this specification and is
   for further study.

9. Contributors

   Rahul Aggarwal of Juniper Networks and George Swallow of Cisco
   Systems provided the initial idea for this specification and
   contributed to its development.

10. Acknowledgments

   Authors would also like to thank Nobo Akiya, Vengada Prasad Govindan,
   Jeff Haas, Wim Henderickx, Gregory Mirsky and Mingui Zhang who have
   greatly contributed to this document.

11. References

11.1. Normative References

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

   [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection
              (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010,
              <https://www.rfc-editor.org/info/rfc5880>.

   [RFC7880] Pignataro, C., Ward, D., Akiya, N., Bhatia, M., and S.
              Pallagatti, "Seamless Bidirectional Forwarding Detection
              (S-BFD)", RFC 7880, DOI 10.17487/RFC7880, July 2016,
              <https://www.rfc-editor.org/info/rfc7880>.

   [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N.,
              Aldrin, S., and M. Chen, "Detecting Multiprotocol Label
              Switched (MPLS) Data-Plane Failures", RFC 8029,
              DOI 10.17487/RFC8029, March 2017,
              <https://www.rfc-editor.org/info/rfc8029>.

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

11.2. Informational References

   [I-D.ietf-bess-mvpn-fast-failover]
              Morin, T., Kebler, R., and G. Mirsky, "Multicast VPN fast
              upstream failover", draft-ietf-bess-mvpn-fast-failover-04
              (work in progress), November 2018.

   [I-D.ietf-bfd-multipoint-active-tail]
             Katz, D., Ward, D., Networks, J., and G. Mirsky, "BFD
             Multipoint Active Tails.", draft-ietf-bfd-multipoint-
             active-tail-10 (work in progress), November 2018.

   [RFC4082] Perrig, A., Song, D., Canetti, R., Tygar, J., and B.
             Briscoe, "Timed Efficient Stream Loss-Tolerant
             Authentication (TESLA): Multicast Source Authentication
             Transform Introduction", RFC 4082, DOI 10.17487/RFC4082,
             June 2005, <https://www.rfc-editor.org/info/rfc4082>.

   [RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection
             (BFD) for Multihop Paths", RFC 5883, DOI 10.17487/RFC5883,
             June 2010, <https://www.rfc-editor.org/info/rfc5883>.

   [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,
             and A. Bierman, Ed., "Network Configuration Protocol
             (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
             <https://www.rfc-editor.org/info/rfc6241>.

Authors' Addresses

   Dave Katz
   Juniper Networks
   1194 N. Mathilda Ave.
   Sunnyvale, California 94089-1206
   USA

   Email: dkatz@juniper.net

   Dave Ward
   Cisco Systems
   170 West Tasman Dr.
   San Jose, California 95134
   USA

   Email: wardd@cisco.com

   Santosh Pallagatti (editor)
   Rtbrick

   Email: santosh.pallagatti@gmail.com

   Greg Mirsky (editor)
   ZTE Corp.

   Email: gregimirsky@gmail.com

Internet Engineering Task Force   D. Katz
Internet-Draft   Juniper Networks
Intended status: Standards Track   D. Ward
Expires: June 1, 2019   Cisco Systems
                                                    S. Pallagatti, Ed.
                                                              Rtbrick
                                                       G. Mirsky, Ed.
                                                            ZTE Corp.
                                                    November 28, 2018

                        BFD Multipoint Active Tails.
                  draft-ietf-bfd-multipoint-active-tail-10

Abstract

   This document describes active tail extensions to the Bidirectional
   Forwarding Detection (BFD) protocol for multipoint networks.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

    This application of BFD is an extension to Multipoint BFD
    [I-D.ietf-bfd-multipoint], which allows tails to notify the head of
    the lack of multipoint connectivity. As a further option, heads can
    request a notification from the tails by means of a polling
    mechanism. Notification to the head can be enabled for all tails, or
    for only a subset of the tails.

    The goal of this application is for the head to reasonably rapidly
    have knowledge of tails that have lost connectivity from the head.

    Since scaling is a primary concern (particularly state explosion
    toward the head), it is required that the head be in control of all
    timing aspects of the mechanism, and that BFD packets from the tails
    to the head not be synchronized.

    Throughout this document, the term "multipoint" is defined as a
    mechanism by which one or more systems receive packets sent by a
    single sender. This specifically includes such things as IP
    multicast and point-to-multipoint MPLS.

    Term "connectivity" in this document is not being used in the context
    of connectivity verification in transport network but as an
    alternative to "continuity", i.e. existence of a path between the
    sender and the receiver.

    This document effectively modifies and adds to Sections 5.12 and 5.13
    of the base BFD multipoint document [I-D.ietf-bfd-multipoint].

2. Terminology and Acronyms

    BFD Bidirectional Forwarding Detection

    c-poll Composite Poll

    m-poll Multipoint Poll

3. Keywords

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

4. Overview

   A head may wish to be alerted to the tails' connectivity (or lack
   thereof), and there are a number of options to achieve that. First,
   if all that is needed is a best-effort failure notification, as
   discussed in Section 5.2.1, the tails can send unsolicited unicast
   BFD Control packets to the head when the path fails, as described in
   Section 6.4.

   If the head wishes to know of the active tails on the multipoint
   path, it may send a multipoint BFD Control packet with the Poll (P)
   bit set, which will induce the tails to return a unicast BFD Control
   packet with the Final (F) bit set (detailed description in
   Section 5.2.2). The head can then create BFD session state for each
   of the tails that have multipoint connectivity. If the head sends
   such a packet on occasion, it can keep track of which tails answer,
   thus providing a more deterministic mechanism for detecting which
   tails fail to respond (implying a loss of multipoint connectivity).
   In this document, this method referenced to as Multipoint Poll
   (m-poll).

   If the head wishes the definite indication of the tails'
   connectivity, it may do all of the above, but if it detects that a
   tail did not answer the previous multipoint poll, it may initiate a
   Demand mode Poll Sequence as a unicast to that tail (detailed
   description in Section 5.2.3). This covers the case where either the
   multipoint poll or the single reply also is lost in transit. If
   desired, the head may Poll one or more tails proactively to track the
   tails' connectivity. In this document this method that combines the
   use of multipoint and unicast polling of tails by the head referenced
   to as Composite Poll (c-poll).

   If the awareness of the state of some nodes is more important for the
   head, in the sense that the head needs to detect the lack of
   multipoint connectivity to a subset of tails at a different rate, the
   head may transmit unicast BFD Polls to that subset of tails. In this
   case, the timing may be independent on a tail-by-tail basis.

   Individual tails may be configured so that they never send BFD
   control packets to the head. Such tails will never be known to the
   head, but will still be able to detect multipoint path failures from
   the head.

5. Operational Scenarios

   It is worth analyzing how this protocol reacts to various scenarios.
   There are three path components present, namely, the multipoint path,
   the forward unicast path (from head to a particular tail), and the

reverse unicast path (from a tail to the head). There are also four
options as to how the head is notified about failures from the tail.
For the different modes described below the setting of new state
variables are given even if these are only introduced later in the
document (see Section 6.3).

5.1. No Head Notification

In this scenario, only the multipoint path is used and none of the
others matter. A failure in the multipoint path will result in the
tail noticing the failure within a detection time, and the head will
remain ignorant of the tail state. This mode emulates the behavior
described in [I-D.ietf-bfd-multipoint]. In this mode,
bfd.SessionType is MultipointTail and the variable bfd.SilentTail
(see Section 6.3.1) MUST be set to 1. If bfd.SessionType is
MultipointHead or MultipointClient bfd.ReportTailDown MUST be set to
0. The head MAY set bfd.RequiredMinRxInterval to zero and thus
suppress tails sending any BFD control packets.

5.2. Head Notification

In these scenarios, the tail sends unsolicited or solicited BFD
packets in response to the detection of a multipoint path failure.
All these scenarios have common settings:

o if bfd.SessionType is MultipointTail, the variable bfd.SilentTail
   (see Section 6.3.1) MUST be set to 0;

o if bfd.SessionType is MultipointHead or MultipointClient
   bfd.ReportTailDown MUST be set to 1;

o the head MUST set bfd.RequiredMinRxInterval to non-zero and thus
   allow tails sending BFD control packets.

5.2.1. Head Notification Without Polling

In this scenario, the tail sends unsolicited BFD packets in response
to the detection of a multipoint path failure. It uses the reverse
unicast path, but not the forward unicast path.

If the multipoint path fails but the reverse unicast path stays up,
the tail will detect the failure within a detection time, and the
head will know about it within one reverse packet time (since the
notification is delayed).

If both the multipoint path and the reverse unicast paths fail, the
tail will detect the failure but the head will remain unaware of it.

5.2.2. Head Notification and Tail Solicitation with Multipoint Polling

   In this scenario, the head sends occasional multipoint Polls in
   addition to (or in lieu of) non-Poll multipoint BFD Control packets,
   expecting the tails to reply with Final. This also uses the reverse
   unicast path, but not the forward unicast path.

   If the multipoint path fails but the reverse unicast path stays up,
   the tail will detect the failure within a detection time, and the
   head will know about it within one reverse packet time (the
   notification is delayed to avoid synchronization of the tails).

   If both the multipoint path and the reverse unicast paths fail, the
   tail will detect the failure but the head will remain unaware of this
   fact.

   If the reverse unicast path fails but the multipoint path stays up,
   the head will see the BFD session fail, but the state of the
   multipoint path will be unknown to the head. The tail will continue
   to receive multipoint data traffic.

   If either the multipoint Poll or the unicast reply is lost in
   transit, the head will see the BFD session fail, but the state of the
   multipoint path will be unknown to the head. The tail will continue
   to receive multipoint data traffic.

5.2.3. Head Notification with Composite Polling

   In this scenario, the head sends occasional multipoint Polls in
   addition to (or in lieu of) non-Poll multipoint BFD control packets,
   expecting the tails to reply with Final. If a tail that had
   previously replied to a multipoint Poll fails to reply (or if the
   head simply wishes to verify tail connectivity), the head issues a
   unicast Poll Sequence to the tail. This scenario makes use of all
   three paths. In this mode for bfd.SessionType of MultipointTail,
   variable bfd.SilentTail (see Section 6.3.1) MUST be set to 0.

   If the multipoint path fails but the two unicast paths stay up, the
   tail will detect the failure within a detection time, and the head
   will know about it within one reverse packet time (since the
   notification is delayed). Note that the reverse packet time may be
   smaller in this case if the head has previously issued a unicast Poll
   (since the tail will not delay transmission of the notification in
   this case).

   If both the multipoint path and the reverse unicast paths fail
   (regardless of the state of the forward unicast path), the tail will
   detect the failure but the head will remain unaware of this fact.

The head will detect a BFD session failure to the tail but cannot
make a determination about the state of the tail's multipoint
connectivity.

If the forward unicast path fails but the reverse unicast path stays
up, the head will detect a BFD session failure to the tail if it
happens to send a unicast Poll sequence, but cannot make a
determination about the state of the tail's multipoint connectivity.
If the multipoint path to the tail fails prior to any unicast Poll
being sent, the tail will detect the failure within a detection time,
and the head will know about it within one reverse packet time (since
the notification is delayed).

If the multipoint path stays up but the reverse unicast path fails,
the head will see the particular MultipointClient session fail if it
happens to send a Poll Sequence, but the state of the multipoint path
will be unknown to the head. The tail will continue to receive
multipoint data traffic.

If the multipoint path and the reverse unicast path both stay up but
the forward unicast path fails, neither side will notice this failure
so long as a unicast Poll Sequence is never sent by the head. If the
head sends a unicast Poll Sequence, the head will detect the failure
in the forward unicast path. The state of the multipoint path will
be determined by multipoint Poll. The tail will continue to receive
multipoint data traffic.

6. Protocol Details

   This section describes the operation of BFD Multipoint active tail in
   detail. This section modifies the section 4 of
   [I-D.ietf-bfd-multipoint] as the following:

   o Section 6.3 introduces new state variables and modifies the usage
      of a few existing ones;

   o Section 6.13 replaces the corresponding sections in the base BFD
      for multipoint networks specification.

6.1. Multipoint Client Session

   If the head is keeping track of some or all of the tails, it has a
   session of type MultipointClient per tail that it cares about. All
   of the MultipointClient sessions for tails on a particular multipoint
   path are associated with the MultipointHead session to which the
   clients are listening. A BFD Poll Sequence may be sent over a
   MultipointClient session to a tail if the head wishes to verify
   connectivity. These sessions receive any BFD Control packets sent by

the tails, and MUST NOT transmit periodic BFD Control packets other
than Poll Sequences (since periodic transmission is always done by
the MultipointHead session). Note that the settings of all BFD
variables in a MultipointClient session for a particular tail
override the corresponding settings in the MultipointHead session.

6.2. Multipoint Client Session Failure

If a MultipointClient session receives a BFD Control packet from the
tail with state Down or AdminDown, the head reliably knows that the
tail has lost multipoint connectivity. If the Detection Time expires
on a MultipointClient session, it is ambiguous as to whether the
multipoint connectivity failed or whether there was a unicast path
problem in one direction or the other, so the head does not reliably
know the tail's state.

6.3. State Variables

BFD Multipoint active tail introduces new state variables and
modifies the usage of a few existing ones defined in section 4.4 of
[I-D.ietf-bfd-multipoint].

6.3.1. New State Variables

A few state variables are added in support of Multipoint BFD active
tail.

    bfd.SilentTail

        If 0, a tail may send packets to the head according to other
        parts of this specification. Setting this to 1 allows tails to
        be provisioned to always be silent, even when the head is
        soliciting traffic from the tails. This can be useful, for
        example, in deployments of a large number of tails when the
        head wishes to track the state of a subset of them. This
        variable MUST be initialized based on configuration. The
        default value MUST be 1.

        This variable is only pertinent when bfd.SessionType is
        MultipointTail and SHOULD NOT be modified after the
        MultipointTail session has been created.

    bfd.ReportTailDown

        Set to 1 if the head wishes tails to notify the head, via
        periodic BFD Control packets, when they see the BFD session
        fail. If 0, the tail will never send periodic BFD Control
        packets, and the head will not be notified of session failures

by the tails. This variable MUST be initialized based on
configuration. The default value MUST be 0.

This variable is only pertinent when bfd.SessionType is
MultipointHead or MultipointClient.

   bfd.UnicastRcvd

Set to 1 if a tail has received a unicast BFD Control packet
from the head while being in Up state. This variable MUST be
set to zero if the session transitions from Up state to some
other state.

This variable MUST be initialized to zero.

This variable is only pertinent when bfd.SessionType is
MultipointTail.

6.3.2. New State Variable Value

A new state variable value being added to:

bfd.SessionType

The type of this session as defined in [RFC7880]. A new value
introduced is:

MultipointClient: A session on the head that tracks the state
of an individual tail, when desirable.

This variable MUST be initialized to the appropriate type when the
session is created, according to the rules in section 4.4 of
[I-D.ietf-bfd-multipoint].

6.3.3. State Variable Initialization and Maintenance

Some state variables defined in section 6.8.1 of [RFC5880] need to be
initialized or manipulated differently depending on the session type.
The values of some of these variables relate to those of the same
variables of a MultipointHead session (see section 4.4.2 of
[I-D.ietf-bfd-multipoint]).

   bfd.LocalDiscr

For session type MultipointClient, this variable MUST always
match the value of bfd.LocalDiscr in the associated
MultipointHead session.

bfd.DesiredMinTxInterval

    For session type MultipointClient, this variable MUST always
    match the value of bfd.DesiredMinTxInterval in the associated
    MultipointHead session.

bfd.RequiredMinRxInterval

    It MAY be set to zero at the head BFD system to suppress
    traffic from the tails. Setting it to zero in the
    MultipointHead session suppresses traffic from all tails, the
    setting in a MultipointClient session suppresses traffic from a
    single tail.

bfd.DemandMode

    This variable MUST be initialized to 1 for session types
    MultipointClient.

bfd.DetectMult

    For session type MultipointClient, this variable MUST always
    match the value of bfd.DetectMult in the associated
    MultipointHead session.

6.4. Controlling Multipoint BFD Options

The state variables defined above are used to choose which
operational options are active.

The most basic form of the operation of BFD in multipoint networks
explained in [I-D.ietf-bfd-multipoint]. In this scenario, BFD
Control packets flow only from the head and no tracking of tail state
at the head is desired. That can be accomplished by setting
bfd.ReportTailDown to 0 in the MultipointHead session (Section 5.1).

If the head wishes to know of active the tails, it sends multipoint
Polls as needed. Previously known tails that don't respond to the
Polls will be detected (as per Section 5.2.2).

If the head wishes to request a notification from the tails when they
lose connectivity, it sets bfd.ReportTailDown to 1 in either the
MultipointHead session (if such notification is desired from all
tails) or in the MultipointClient session (if notification is desired
from a particular tail). Note that the setting of this variable in a
MultipointClient session for a particular tail overrides the setting
in the MultipointHead session.

If the head wishes to verify the state of a tail on an ongoing basis,
it sends a Poll Sequence from the MultipointClient session associated
with that tail as needed. This has the effect of eliminating the
initial delay, described in Section 6.13.3, that the tail would
otherwise insert prior to transmission of the packet thus the head
may have notification of the session failure more quickly when
comparing with use of m-poll.

If a tail wishes to operate silently (sending no BFD Control packets
to the head) it sets bfd.SilentTail to 1 in the MultipointTail
session. This allows a tail to be silent independent of the settings
on the head.

6.5. State Machine

Though the state transitions for the state machine, as defined in
section 5.5 of [I-D.ietf-bfd-multipoint], for a session type
MultipointHead are only administratively driven, the state machine
for a session of type MultipointClient is the same and the diagram is
applicable.

6.6. Session Establishment

If BFD Control packets are received at the head, they are
demultiplexed to sessions of type MultipointClient, which represent
the set of tails that the head is interested in tracking. These
sessions will typically also be established dynamically based on the
receipt of BFD Control packets. The head has broad latitude in
choosing which tails to track, if any, without affecting the basic
operation of the protocol. The head directly controls whether or not
tails are allowed to send BFD Control packets back to the head by
setting bfd.RequiredMinRxInterval to zero in a MultipointHead or a
MultipointClient session.

6.7. Discriminators and Packet Demultiplexing

When the tails send BFD Control packets to the head from the
MultipointTail session, the contents of Your Discriminator (the
discriminator received from the head) will not be sufficient for the
head to demultiplex the packet, since the same value will be received
from all tails on the multicast tree. In this case, the head MUST
demultiplex packets based on the source address and the value of Your
Discr, which together uniquely identify the tail and the multipoint
path.

When the head sends unicast BFD Control packets to a tail from a
MultipointClient session, the value of Your Discriminator will be

   valid, and the tail MUST demultiplex the packet based solely on Your
   Discr.

6.8. Controlling Tail Packet Transmission

   As the fan-in from the tails to the head may be very large, it is
   critical that the flow of BFD Control packets from the tails is
   controlled.

   The head always operates in Demand mode. This means that no tail
   will send an asynchronous BFD Control packet as long as the session
   is Up.

   The value of Required Min Rx Interval received by a tail in a unicast
   BFD Control packet, if any, always takes precedence over the value
   received in Multipoint BFD Control packets. This allows the packet
   rate from individual tails to be controlled separately as desired by
   sending a BFD Control packet from the corresponding MultipointClient
   session. This also eliminates the random delay, as discussed in
   Section 6.13.3, prior to transmission from the tail that would
   otherwise be inserted, reducing the latency of reporting a failure to
   the head.

   If the head wishes to suppress traffic from the tails when they
   detect a session failure, it MAY set bfd.RequiredMinRxInterval to
   zero, which is a reserved value that indicates that the sender wishes
   to receive no periodic traffic. This can be set in the
   MultipointHead session (suppressing traffic from all tails) or it can
   be set in a MultipointClient session (suppressing traffic from only a
   single tail).

   Any tail may be provisioned to never send *any* BFD Control packets
   to the head by setting bfd.SilentTail to 1. This provides a
   mechanism by which only a subset of tails reports their session
   status to the head.

6.9. Soliciting the Tails

   If the head wishes to know of the active tails, the MultipointHead
   session can send a BFD Control packet as specified in Section 6.13.3,
   with the Poll (P) bit set to 1. This will cause all of the tails to
   reply with a unicast BFD Control Packet, randomized across one packet
   interval.

   The decision as to when to send a multipoint Poll is outside the
   scope of this specification. However, it MUST NOT be sent more often
   than the regular multipoint BFD Control packet. Since the tail will

treat a multipoint Poll like any other multipoint BFD Control packet,
Polls may be sent in lieu of non-Poll packets.

Soliciting the tails also starts the Detection Timer for each of the
associated MultipointClient sessions, which will cause those sessions
to time out if the associated tails do not respond.

Note that for this mechanism to work properly, the Detection Time
(which is equal to bfd.DesiredMinTxInterval) MUST be greater than the
round trip time of BFD Control packets from the head to the tail (via
the multipoint path) and back (via a unicast path). See Section 6.11
for more details.

6.10. Verifying Connectivity to Specific Tails

   If the head wishes to verify connectivity to a specific tail, the
   corresponding MultipointClient session can send a BFD Poll Sequence
   to said tail. This might be done in reaction to the expiration of
   the Detection Timer (the tail didn't respond to a multipoint Poll),
   or it might be done on a proactive basis.

   The interval between transmitted packets in the Poll Sequence MUST be
   calculated as specified in the base BFD specification [RFC5880] (the
   greater of bfd.DesiredMinTxInterval and bfd.RemoteMinRxInterval).

   The value transmitted in Required Min RX Interval will be used by the
   tail (rather than the value received in any multipoint packet) when
   it transmits BFD Control packets to the head notifying it of a
   session failure and the transmitted packets will not be delayed.
   This value can potentially be set much lower than in the multipoint
   case, in order to speed up a notification to the head, since the
   value will be used only by the single tail. This value (and the lack
   of delay) are "sticky", in that once the tail receives it, it will
   continue to use it indefinitely. Therefore, if the head no longer
   wishes to single out the tail, it SHOULD reset the timer to the
   default by sending a Poll Sequence with the same value of Required
   Min Rx Interval as is carried in the multipoint packets, or it MAY
   reset the tail session by sending a Poll Sequence with state
   AdminDown (after the completion of which the session will come back
   up).

   Note that a failure of the head to receive a response to a Poll
   Sequence does not necessarily mean that the tail has lost multipoint
   connectivity, though a reply to a Poll Sequence does reliably
   indicate connectivity or lack thereof (by virtue of the tail's state
   not being Up in the BFD Control packet).

6.11. Detection Times

   MultipointClient sessions at the head are always in the Demand mode,
   and as such only care about detection time in two cases. First, if a
   Poll Sequence is being sent on a MultipointClient session, the
   detection time on this session is calculated according to the base
   BFD specification [RFC5880], that is, the transmission interval
   multiplied by bfd.DetectMult. Second, when a multipoint Poll is sent
   to solicit tail replies, the detection time on all associated
   MultipointClient sessions that aren't currently sending Poll
   Sequences is set to a value greater than or equal to
   bfd.RequiredMinRxInterval (one packet time). This value can be made
   arbitrarily large in order to ensure that the detection time is
   greater than the round trip time of a BFD Control packet between the
   head and the tail with no ill effects, other than delaying the
   detection of unresponsive tails. Note that a detection time
   expiration on a MultipointClient session at the head, while
   indicating a BFD session failure, cannot be construed to mean that
   the tail is not hearing multipoint packets from the head.

6.12. MultipointClient Down/AdminDown Sessions

   If the MultipointHead session is in Down/AdminDown state (which only
   happens administratively), all associated MultipointClient sessions
   SHOULD be destroyed as they are superfluous.

   If a MultipointClient session goes down due to the receipt of an
   unsolicited BFD Control packet from the tail with state Down or
   AdminDown (not in response to a Poll), and tail connectivity
   verification is not being done, the session MAY be destroyed. If
   verification is desired, the session SHOULD send a Poll Sequence and
   the session SHOULD be maintained.

   If the tail replies to a Poll Sequence with state Down or AdminDown,
   it means that the tail session is definitely down. In this case, the
   session MAY be destroyed.

   If the Detection Time expires on a MultipointClient session (meaning
   that the tail did not reply to a Poll Sequence) the session MAY be
   destroyed.

6.13. Base BFD for Multipoint Networks Specification Text Replacement

   The following sections are meant to extend the corresponding sections
   in the base BFD for Multipoint Networks specification
   [I-D.ietf-bfd-multipoint].

6.13.1. Reception of BFD Control Packets

   The following procedure modifies parts of Section 5.13.1 of
   [I-D.ietf-bfd-multipoint].

   When a BFD Control packet is received, the procedure defined in
   Section 5.13.1 of [I-D.ietf-bfd-multipoint] MUST be followed, in the
   order specified. If the packet is discarded according to these
   rules, processing of the packet MUST cease at that point. In
   addition to that, if tail tracking is desired by the head, the
   following procedure MUST be applied.

      If bfd.SessionType is MultipointTail

         If bfd.UnicastRcvd is 0 or the M bit is clear, set
         bfd.RemoteMinRxInterval to the value of Required Min RX
         Interval.

         If the M bit is clear, set bfd.UnicastRcvd to 1.

      Else (not MultipointTail)

         Set bfd.RemoteMinRxInterval to the value of Required Min RX
         Interval.

      If the Poll (P) bit is set, and bfd.SilentTail is zero, send a BFD
      Control packet to the remote system with the Poll (P) bit clear,
      and the Final (F) bit set (see Section 6.13.3).

6.13.2. Demultiplexing BFD Control Packets

   This section is part of the addition to Section 5.13.2 of
   [I-D.ietf-bfd-multipoint], separated for clarity.

      If Multipoint (M) bit is clear

         If the Your Discriminator field is nonzero

            Select a session based on the value of Your Discriminator.
            If no session is found, the packet MUST be discarded.

         If bfd.SessionType is MultipointHead

            Find a MultipointClient session grouped to this
            MultipointHead session, based on the source address and
            the value of Your Discriminator. If a session is found
            and is not MultipointClient, the packet MUST be
            discarded. If no session is found, a new session of type

MultipointClient MAY be created, or the packet MAY be
discarded. This choice is outside the scope of this
specification.

If bfd.SessionType is not MultipointClient, the packet
MUST be discarded.

6.13.3. Transmitting BFD Control Packets

A system MUST NOT periodically transmit BFD Control packets if
bfd.SessionType is MultipointClient and a Poll Sequence is not being
transmitted.

If bfd.SessionType value is MultipointTail and the periodic
transmission of BFD Control packets is just starting (due to Demand
mode not being active on the remote system), the first packet to be
transmitted MUST be delayed by a random amount of time between zero
and (0.9 * bfd.RemoteMinRxInterval).

If a BFD Control packet is received with the Poll (P) bit set to 1,
the receiving system MUST transmit a BFD Control packet with the Poll
(P) bit clear and the Final (F) bit, without respect to the
transmission timer or any other transmission limitations, without
respect to the session state, and without respect to whether Demand
mode is active on either system. A system MAY limit the rate at
which such packets are transmitted. If rate limiting is in effect,
the advertised value of Desired Min TX Interval MUST be greater than
or equal to the interval between transmitted packets imposed by the
rate limiting function. If the Multipoint (M) bit is set in the
received packet, the packet transmission MUST be delayed by a random
amount of time between zero and (0.9 * bfd.RemoteMinRxInterval).
Otherwise, the packet MUST be transmitted as soon as practicable.

A system MUST NOT set the Demand (D) bit if bfd.SessionType is
MultipointClient unless bfd.DemandMode is 1, bfd.SessionState is Up,
and bfd.RemoteSessionState is Up.

Content of the transmitted packet MUST be as explained in section
5.13.3 of [I-D.ietf-bfd-multipoint].

7. Assumptions

If head notification is to be used, it is assumed that a multipoint
BFD packet encapsulation contains enough information so that a tail
can address a unicast BFD packet to the head.

If head notification is to be used, it is assumed that is that there
is bidirectional unicast communication available (at the same

protocol layer within which BFD is being run) between the tail and head.

For the head to know reliably that a tail has lost multipoint connectivity, the unicast paths in both directions between that tail and the head must remain operational when the multipoint path fails. It is thus desirable that unicast paths not share fate with the multipoint path to the extent possible if the head wants more definite knowledge of the tail state.

Since the normal BFD three-way handshake is not used in this application, a tail transitioning from state Up to Down and back to Up again may not be reliably detected at the head.

8. Operational Considerations

Section 7 of [RFC5880] includes the requirements for implementation of a congestion control mechanism when BFD is used across multiple hops, and the mechanism to use congestion detection to reduce the amount of BFD packets the system generates. These requirements are also applicable to this specification. When this specification used in the mode with no head notifications by tails, as discussed in Section 5.1, the head MUST limit the packet transmission rate to not higher than one BFD packet per second (Section 6 [I-D.ietf-bfd-multipoint]). When the BFD uses one of notification by tails to head mechanisms described in Section 5.2, Min RX Interval can be used by the tail to control the packet transmission rate of the head. The exact mechanism of processing changes in the Min RX Interval value in the received from the tail response to multicast or unicast Poll BFD packet is outside the scope of this document.

As noted in Section 7 [RFC5880], "any mechanism that increases the transmit or receive intervals will increase the Detection Time for the session".

9. IANA Considerations

This document has no actions for IANA.

10. Security Considerations

The same security considerations as those described in [RFC5880] and [I-D.ietf-bfd-multipoint] apply to this document.

Additionally, implementations that create MultpointClient sessions dynamically upon receipt of BFD Control packet from a tail MUST implement protective measures to prevent a number of MultipointClient

sessions being created growing out of control. Below are listed some
points to be considered in such implementations.

   When the number of MultipointClient sessions exceeds the number of
   expected tails, then the implementation should generate an alarm
   to users to indicate the anomaly.

   The implementation should have a reasonable upper bound on the
   number of MultipointClient sessions that can be created, with the
   upper bound potentially being computed based on the number of
   multicast streams that the system is expecting.

This specification does not raise any additional security issues
beyond those of the specifications referred to in the list of
normative references.

11. Contributors

Rahul Aggarwal of Juniper Networks and George Swallow of Cisco
Systems provided the initial idea for this specification and
contributed to its development.

12. Acknowledgments

Authors would also like to thank Nobo Akiya, Vengada Prasad Govindan,
Jeff Haas, Wim Henderickx and Mingui Zhang who have greatly
contributed to this document.

13. Normative References

[I-D.ietf-bfd-multipoint]
          Katz, D., Ward, D., Networks, J., and G. Mirsky, "BFD for
          Multipoint Networks", draft-ietf-bfd-multipoint-18 (work
          in progress), June 2018.

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

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection
          (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010,
          <https://www.rfc-editor.org/info/rfc5880>.

[RFC7880] Pignataro, C., Ward, D., Akiya, N., Bhatia, M., and S.
          Pallagatti, "Seamless Bidirectional Forwarding Detection
          (S-BFD)", RFC 7880, DOI 10.17487/RFC7880, July 2016,
          <https://www.rfc-editor.org/info/rfc7880>.

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

Authors' Addresses

   Dave Katz
   Juniper Networks
   1194 N. Mathilda Ave.
   Sunnyvale, California 94089-1206
   USA

   Email: dkatz@juniper.net

   Dave Ward
   Cisco Systems
   170 West Tasman Dr.
   San Jose, California 95134
   USA

   Email: wardd@cisco.com

   Santosh Pallagatti (editor)
   Rtbrick

   Email: santosh.pallagatti@gmail.com

   Greg Mirsky (editor)
   ZTE Corp.

   Email: gregimirsky@gmail.com

Network Working Group   M. Jethanandani
Internet-Draft   VMware
Intended status: Standards Track   A. Mishra
Expires: May 15, 2019   SES Networks
                                                               A. Saxena
                                                       Ciena Corporation
                                                               M. Bhatia
                                                                   Nokia
                                                       November 11, 2018

                        Optimizing BFD Authentication
                    draft-ietf-bfd-optimizing-authentication-07

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 BCP 14 [RFC2119]
   [RFC8174] when, and only when, they appear in all capitals, as shown
   here.

Copyright Notice

Table of Contents

1. Introduction

   Authenticating every BFD [RFC5880] packet with a Simple Password, or
   with a MD5 Message-Digest Algorithm [RFC1321] , or Secure Hash
   Algorithm (SHA-1) algorithms is computationally intensive process,
   making it difficult if not impossible to authenticate every packet -
   particularly at faster rates. Also, 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 overhead, 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. Rest of the frames can be
   transmitted and received without authentication enabled. Most frames
   that are transmitted and received have no state change associated
   with them. Limiting authentication to frames that affect a BFD
   session state allows 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. To detect a Man In the Middle (MITM)
attack, it is also proposed that a non-state change frame be
authenticated occasionally. The interval of this non-state change
frame can be configured depending on the detect multiplier and the
capability of the system. As an example, this could be equal to the
detect multiplier number of packets.

The rest of the document is structured as follows. Section 2 talks
about the changes to authentication mode as described in BFD
[RFC5880]. Section 3 goes into the details of the new Authentication
TLV.

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
enabled, every packet must have Authentication Bit set and the
associated Authentication TLV appended. 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 authenticating every packet, BFD peers
are configured for which frames need to be authenticated, and
authenticate only those frames. Rest of the frames can be
transmitted and received without authentication. For example, the
two ends can be configured such that BFD frames that indicate a state
change should be authenticated and enable authentication on those
frames only. If the two ends have previously been configured as
such, but at least one side decides not to authenticate a state
change frame, then the BFD session will fail to come up.

This proposal outlines which frames need to be authenticated (carry
the A-bit), and which frames can be transmitted or received without
authentication enabled. A frame that fails authentication is
discarded, or a frame that was supposed to be authenticated, but was
not, e.g. a state-change frame, is discarded. However, there is no
change to the state machine for BFD, as the decision of a state
change is still decided by how many valid consecutive frames were
received, authenticated or otherwise.

The state changes for which authentication is being suggested
include:

```
        Read   : On state change from <column> to <row>
        Auth   : Authenticate frame
        NULL   : No Authentication. Use NULL AUTH TLV.
        n/a    : Invalid state transition.
        Select : Most frames NULL AUTH. Selective (periodic)
                 frames authenticated.
```

	DOWN	INIT	UP	POLL	DEMAND
DOWN	NULL	Auth	Auth	Auth	Auth
INIT	Auth	NULL	Auth	Auth	Auth
UP	Auth	n/a	Select	Auth	Auth
POLL	Auth	n/a	Auth	Auth	Auth
DEMAND	Auth	Auth	Auth	Auth	Auth

Optimized Authentication Map

All frames already carry the sequence number. The NULL AUTH frames
MUST contain the TLV specified in Section 3. This enables a
monotonically increasing sequence number to be carried in each frame,
and prevents man-in-the-middle from capturing and replaying the same
frame again. Since all frames still carry a sequence number, the
logic for sequence number maintenance remains unchanged from
[RFC5880]. If at a later time, a different scheme is adopted for
changing sequence number, this method can use the updated scheme
without any impact.

Most frames transmitted on a BFD session are BFD CC UP frames.
Authenticating a small subset of these frames, for example, a detect
multiplier number of packets per configured period, significantly
reduces the computational demand for the system while maintaining
security of the session across the configured authentication periods.
A minimum of Detect Multiplier packets MUST be transmitted per
configured periodic authentication interval. This ensures that the
BFD session should see at least one authenticated packet during that
interval.

3. NULL Auth TLV

   This section describes a new Authentication TLV as:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Auth Type   |    Auth Len   |  Auth Key ID  |    Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
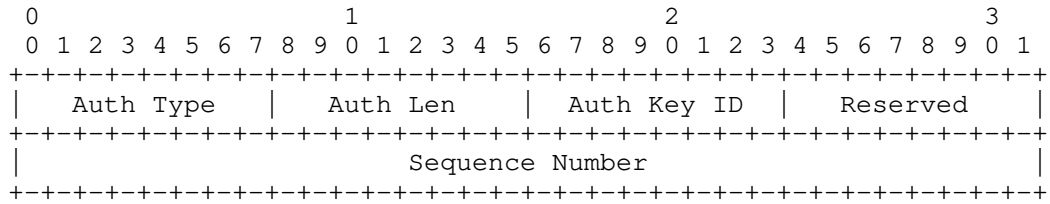
                           NULL Auth TLV

   where:

   Auth Type: The Authentication Type, which in this case is TBD (NULL
   Auth TLV, to be assigned by IANA)

   Auth Len: The length of the NULL Auth TLV, in bytes i.e. 8 bytes

   Auth Key ID: The authentication key ID in use for this packet. Must
   be set to zero.

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

   Sequence Number: The sequence number for this packet. Implementation
   may use sequence numbers as defined in [RFC5880], or secure sequence
   numbers as defined in [I-D.ietf-bfd-secure-sequence-numbers].

   The NULL Auth TLV must be used for all frames that are not
   authenticated. This protects against replay-attacks by allowing the
   session to maintain an incrementing sequence number for all frames
   (authenticated and un-authenticated).

   In the future, if a new scheme is adopted for changing the sequence
   number, this method can adopt the new scheme without any impact.

4. IANA Considerations

   This document requests an update to the registry titled "BFD
   Authentication Types". IANA is requested to to assign a new BFD Auth
   Type for "NULL Auth TLV" (see Section 3).

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

5. 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.

6. References

6.1. Normative References

   [I-D.ietf-bfd-secure-sequence-numbers]
              Jethanandani, M., Agarwal, S., Mishra, A., Saxena, A., and
              A. DeKok, "Secure BFD Sequence Numbers", draft-ietf-bfd-
              secure-sequence-numbers-02 (work in progress), May 2018.

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

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

6.2. Informative References

   [RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321,
              DOI 10.17487/RFC1321, April 1992,
              <https://www.rfc-editor.org/info/rfc1321>.

   [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection
              (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010,
              <https://www.rfc-editor.org/info/rfc5880>.

   [RFC6151] Turner, S. and L. Chen, "Updated Security Considerations
              for the MD5 Message-Digest and the HMAC-MD5 Algorithms",
              RFC 6151, DOI 10.17487/RFC6151, March 2011,
              <https://www.rfc-editor.org/info/rfc6151>.

   [RFC6194] Polk, T., Chen, L., Turner, S., and P. Hoffman, "Security
              Considerations for the SHA-0 and SHA-1 Message-Digest
              Algorithms", RFC 6194, DOI 10.17487/RFC6194, March 2011,
              <https://www.rfc-editor.org/info/rfc6194>.

   [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
   VMware
   USA

   Email: mjethanandani@gmail.com

   Ashesh Mishra
   SES Networks

   Email: mishra.ashesh@gmail.com

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

   Email: ankurpsaxena@gmail.com

   Manav Bhatia
   Nokia
   Bangalore
   India

   Email: manav.bhatia@nokia.com

Network Working Group   R. Rahman, Ed.
Internet-Draft   Cisco Systems
Intended status: Standards Track   L. Zheng, Ed.
Expires: February 2, 2019   Huawei Technologies
                                                     M. Jethanandani, Ed.
                                                       Xoriant Corporation
                                                            S. Pallagatti
                                                                  Rtbrick
                                                               G. Mirsky
                                                          ZTE Corporation
                                                           August 1, 2018

          YANG Data Model for Bidirectional Forwarding Detection (BFD)
                         draft-ietf-bfd-yang-17

Abstract

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

   The YANG modules in this document conform to the Network Management
   Datastore Architecture (NMDA).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   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, a.k.a. BFD for IPv4 and IPv6 [RFC5881]

   2) Two systems connected via multiple hops as described in BFD for
   Multiple Hops. [RFC5883]

   3) Two systems connected via MPLS Label Switched Paths (LSPs) as
   described in BFD for MPLS LSP [RFC5884]

   4) Two systems connected via a Link Aggregation Group (LAG) interface
   as described in BFD on LAG Interfaces [RFC7130]

5) Two systems connected via pseudowires (PWs), this is known as
Virtual Circuit Connectivity Verification (VCCV) as described in BFD
for PW VCCV [RFC5885]. This is not addressed in this document.

BFD typically does not operate on its own. Various control
protocols, also known as BFD clients, use the services provided by
BFD for their own operation as described in Generic Application of
BFD [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 support sub-second failure
detection, e.g. OSPF and IS-IS.

The YANG modules in this document conform to the Network Management
Datastore Architecture (NMDA) [RFC8342]. This means that the data
models do not have separate top-level or sibling containers for
configuration and operational state data.

1.1. 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 BCP 14 [RFC2119]
[RFC8174] when, and only when, they appear in all capitals, as shown
here.

1.2. Tree Diagrams

This document uses the graphical representation of data models
defined in [RFC8340].

2. Design of the Data Model

Since BFD is used for liveliness detection of various forwarding
paths, there is no uniform key to identify a BFD session, and so the
BFD data model is split in multiple YANG modules where each module
corresponds to one type of forwarding path. For example, BFD for IP
single-hop is in one YANG module and BFD for MPLS-TE is in another
YANG module. The main difference between these modules is how a BFD
session is uniquely identified, i.e the key for the list containing
the BFD sessions for that forwarding path. To avoid duplication of
BFD definitions, we have common types and groupings which are used by
all the modules.

A new control-plane protocol "bfdv1" is defined and a "bfd" container
is created under control-plane-protocol as specified in "A YANG Data
Model for Routing Management (NMDA Version)" [RFC8349]. This new
"bfd" container is augmented by all the YANG modules for their
respective specific information:

   1. ietf-bfd-ip-sh.yang augments "/routing/control-plane-protocols/
       control-plane-protocol/bfd/" with the "ip-sh" container for BFD
       sessions over IP single-hop.

   2. ietf-bfd-ip-mh.yang augments "/routing/control-plane-protocols/
       control-plane-protocol/bfd/" with the "ip-mh" container for BFD
       sessions over IP multi-hop.

   3. ietf-bfd-lag.yang augments "/routing/control-plane-protocols/
       control-plane-protocol/bfd/" with the "lag" container for BFD
       sessions over LAG.

   4. ietf-bfd-mpls.yang augments "/routing/control-plane-protocols/
       control-plane-protocol/bfd/" with the "mpls" container for BFD
       over MPLS LSPs.

   5. ietf-bfd-mpls-te.yang augments "/routing/control-plane-protocols/
       control-plane-protocol/bfd/" with the "mpls-te" container for BFD
       over MPLS-TE.

   BFD can operate in the following contexts:

   1. At the network device level

   2. In Logical Network Elements as described in YANG Logical Network
       Element [I-D.ietf-rtgwg-lne-model]

   3. In Network Instances as described in YANG Logical Network Element
       [I-D.ietf-rtgwg-ni-model]

   When used at the network device level, the BFD YANG model is used
   "as-is". When the BFD YANG model is used in a Logical Network
   Element or in a Network Instance, then the BFD YANG model augments
   the mounted routing model for the Logical Network Element or the
   Network Instance.

2.1. Design of Configuration Model

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

   BFD clients are applications that use BFD for fast detection of
   failures. Some implementations have BFD session configuration under
   the BFD clients. For example, BFD session configuration under
   routing applications such as OSPF, IS-IS, BGP etc. Other

implementations have BFD session configuration centralized under BFD,
i.e. outside the multiple BFD clients.

The BFD parameters of interest to a BFD client are mainly the
multiplier and interval(s) since those parameters impact the
convergence time of the BFD clients when a failure occurs. Other
parameters such as BFD authentication are not specific to the
requirements of the BFD client. Ideally all configuration should be
centralized under BFD. However, this is a problem for clients of BFD
which auto-discover their peers. For example, IGPs do not have the
peer address configured, instead the IGP is enabled on an interface
and the IGP peers are auto-discovered. So for an operator to
configure BFD to an IGP peer, the operator would first have to
determine the peer addresses. And when a new peer is discovered, BFD
configuration would need to be added. To avoid this issue, we define
grouping client-cfg-parms in Section 2.13 for BFD clients to
configure BFD: this allows BFD clients such as the IGPs to have
configuration (multiplier and intervals) for the BFD sessions they
need. For example, when a new IGP peer is discovered, the IGP would
create a BFD session to the newly discovered peer and similarly when
an IGP peer goes away, the IGP would remove the BFD session to that
peer. The mechanism how the BFD sessions are created and removed by
the BFD clients is outside the scope of this document, but typically
this would be done by use of an API implemented by the BFD module on
the system. For BFD clients which create BFD sessions via their own
configuration, authentication parameters (if required) are still
specified in BFD.

2.1.1. Common BFD configuration parameters

   The basic BFD configuration parameters are:

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

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

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

   Although BFD [RFC5880] allows for different values for transmit and
   receive intervals, some implementations allow users to specify just
   one interval which is used for both transmit and receive intervals or
   separate values for transmit and receive intervals. The BFD YANG

model supports this: there is a choice between "min-interval", used
for both transmit and receive intervals, and "desired-min-tx-
interval" and "required-min-rx-interval". This is supported via a
grouping which is used by the YANG modules for the various forwarding
paths.

For BFD authentication we have:

key-chain
        This is a reference to key-chain defined in YANG Data Model
        for Key Chains [RFC8177]. The keys, cryptographic
        algorithms, key lifetime etc are all defined in the key-chain
        model.

meticulous
        This enables meticulous mode as per BFD [RFC5880].

2.1.2. Single-hop IP

For single-hop IP, there is an augment of the "bfd" data node in
Section 2. The "ip-sh" node contains a list of IP single-hop
sessions where each session is uniquely identified by the interface
and destination address pair. For the configuration parameters we
use what is defined in Section 2.1.1. The "ip-sh" node also contains
a list of interfaces, this is used to specify authentication
parameters for BFD sessions which are created by BFD clients, see
Section 2.1.

[RFC5880] and [RFC5881] do not specify whether echo function is
continuous or on demand. Therefore the mechanism used to start and
stop echo function is implementation specific and should be done by
augmentation:

   1) Configuration. This is suitable for continuous echo function.
   An example is provided in Appendix A.

   2) RPC. This is suitable for on-demand echo function.

2.1.3. Multihop IP

For multihop IP, there is an augment of the "bfd" data node in
Section 2.

Because of multiple paths, there could be multiple multihop IP
sessions between a source and a destination address. We identify
this as a "session-group". The key for each "session-group" consists
of:

source address
        Address belonging to the local system as per BFD for Multiple
        Hops [RFC5883]

destination address
        Address belonging to the remote system as per BFD for
        Multiple Hops [RFC5883]

For the configuration parameters we use what is defined in
Section 2.1.1

Here are some extra parameters:

tx-ttl
        TTL of outgoing BFD control packets.

rx-ttl
        Minimum TTL of incoming BFD control packets.

2.1.4. MPLS Traffic Engineering Tunnels

For MPLS-TE tunnels, BFD is configured under the MPLS-TE tunnel since
the desired failure detection parameters are a property of the MPLS-
TE tunnel. This is achieved by augmenting the MPLS-TE data model in
YANG Data Model for TE Topologies [I-D.ietf-teas-yang-te]. For BFD
parameters which are specific to the TE application, e.g. whether to
tear down the tunnel in the event of a BFD session failure, these
parameters will be defined in the YANG model of the MPLS-TE
application.

On top of the usual BFD parameters, we have the following per MPLS-TE
tunnel:

encap
        Encapsulation for the BFD packets: choice between IP, G-ACh
        and IP with G-ACh as per MPLS Generic Associated Channel
        [RFC5586]

For general MPLS-TE data, "mpls-te" data node is added under the
"bfd" node in Section 2. Since some MPLS-TE tunnels are uni-
directional there is no MPLS-TE configuration for these tunnels on
the egress node (note that this does not apply to bi-directional
MPLS-TP tunnels). The BFD parameters for the egress node are added
under "mpls-te".

2.1.5. MPLS Label Switched Paths

   Here we address MPLS LSPs whose FEC is an IP address. The "bfd" node
   in Section 2 is augmented with "mpls" which contains a list of
   sessions uniquely identified by an IP prefix. Because of multiple
   paths, there could be multiple MPLS sessions to an MPLS FEC. We
   identify this as a "session-group".

   Since these LSPs are uni-directional there is no LSP configuration on
   the egress node.

   The BFD parameters for the egress node are added under "mpls".

2.1.6. Link Aggregation Groups

   Per BFD on LAG Interfaces [RFC7130], configuring BFD on LAG consists
   of having micro-BFD sessions on each LAG member link. Since the BFD
   parameters are an attribute of the LAG, they should be under the LAG.
   However there is no LAG YANG model which we can augment. So a "lag"
   data node is added to the "bfd" node in Section 2, the configuration
   is per-LAG: we have a list of LAGs. The destination IP address of
   the micro-BFD sessions is configured per-LAG and per address-family
   (IPv4 and IPv6)

2.2. Design of Operational State Model

   The operational state model contains both the overall statistics of
   BFD sessions running on the device and the per session operational
   information.

   The overall statistics of BFD sessions consist of number of BFD
   sessions, number of BFD sessions up etc. This information is
   available globally (i.e. for all BFD sessions) under the "bfd" node
   in Section 2 and also per type of forwarding path.

   For each BFD session, mainly three categories of operational state
   data 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. The
   second category includes a BFD session running information, e.g. the
   remote BFD state and the 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 the session, e.g. when the session
   transitioned up/down and how long it has been in that state.

For some path types, there may be more than 1 session on the virtual
path to the destination. For example, with IP multihop and MPLS
LSPs, there could be multiple BFD sessions from the source to the
same destination to test the various paths (ECMP) to the destination.
This is represented by having multiple "sessions" under each
"session-group".

2.3. Notifications

This YANG model defines notifications to inform end-users of
important events detected during the protocol operation. Pair of
local and remote discriminator identifies a BFD session on local
system. Notifications also give more important details about BFD
sessions; e.g. new state, time in previous state, network-instance
and the reason that the BFD session state changed. The notifications
are defined for each type of forwarding path but use groupings for
common information.

2.4. RPC Operations

None.

2.5. BFD top level hierarchy

At the "bfd" node under control-plane-protocol, there is no
configuration data, only operational state data. The operational
state data consist of overall BFD session statistics, i.e. for BFD on
all types of forwarding paths.

```
module: ietf-bfd
  augment /rt:routing/rt:control-plane-protocols
          /rt:control-plane-protocol:
    +--rw bfd
       +--ro summary
          +--ro number-of-sessions?            yang:gauge32
          +--ro number-of-sessions-up?         yang:gauge32
          +--ro number-of-sessions-down?       yang:gauge32
          +--ro number-of-sessions-admin-down? yang:gauge32
```

2.6. BFD IP single-hop hierarchy

An "ip-sh" node is added under "bfd" node in control-plane-protocol.
The configuration and operational state data for each BFD IP single-
hop session is under this "ip-sh" node.

```
module: ietf-bfd-ip-sh
```

```
      augment /rt:routing/rt:control-plane-protocols
              /rt:control-plane-protocol/bfd:bfd:
       +--rw ip-sh
          +--ro summary
          |  +--ro number-of-sessions?              yang:gauge32
          |  +--ro number-of-sessions-up?           yang:gauge32
          |  +--ro number-of-sessions-down?         yang:gauge32
          |  +--ro number-of-sessions-admin-down?   yang:gauge32
          +--rw sessions
          |  +--rw session* [interface dest-addr]
          |     +--rw interface                    if:interface-ref
          |     +--rw dest-addr                    inet:ip-address
          |     +--rw source-addr?                 inet:ip-address
          |     +--rw local-multiplier?            multiplier
          |     +--rw (interval-config-type)?
          |     |  +--:(tx-rx-intervals)
          |     |  |  +--rw desired-min-tx-interval?    uint32
          |     |  |  +--rw required-min-rx-interval?   uint32
          |     |  +--:(single-interval) {single-minimum-interval}?
          |     |     +--rw min-interval?               uint32
          |     +--rw demand-enabled?              boolean
          |     |      {demand-mode}?
          |     +--rw admin-down?                  boolean
          |     +--rw authentication! {authentication}?
          |     |  +--rw key-chain?    kc:key-chain-ref
          |     |  +--rw meticulous?   boolean
          |     +--ro path-type?                   identityref
          |     +--ro ip-encapsulation?            boolean
          |     +--ro local-discriminator?         discriminator
          |     +--ro remote-discriminator?        discriminator
          |     +--ro remote-multiplier?           multiplier
          |     +--ro demand-capability?           boolean
          |     |      {demand-mode}?
          |     +--ro source-port?                 inet:port-number
          |     +--ro dest-port?                   inet:port-number
          |     +--ro session-running
          |     |  +--ro session-index?            uint32
          |     |  +--ro local-state?              state
          |     |  +--ro remote-state?             state
          |     |  +--ro local-diagnostic?
          |     |  |      iana-bfd-types:diagnostic
          |     |  +--ro remote-diagnostic?
          |     |  |      iana-bfd-types:diagnostic
          |     |  +--ro remote-authenticated?     boolean
          |     |  +--ro remote-authentication-type?
          |     |  |      iana-bfd-types:auth-type {authentication}?
          |     |  +--ro detection-mode?           enumeration
          |     |  +--ro negotiated-tx-interval?   uint32
```

```
            |    | +--ro negotiated-rx-interval?        uint32
            |    | +--ro detection-time?                uint32
            |    | +--ro echo-tx-interval-in-use?       uint32
            |    |           {echo-mode}?
            |    +--ro session-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 down-count?                       yang:counter32
            |       +--ro admin-down-count?                 yang:counter32
            |       +--ro receive-packet-count?             yang:counter64
            |       +--ro send-packet-count?                yang:counter64
            |       +--ro receive-invalid-packet-count?     yang:counter64
            |       +--ro send-failed-packet-count?         yang:counter64
            +--rw interfaces* [interface]
               +--rw interface          if:interface-ref
               +--rw authentication! {authentication}?
                  +--rw key-chain?    kc:key-chain-ref
                  +--rw meticulous?   boolean

    notifications:
      +---n singlehop-notification
         +--ro local-discr?                 discriminator
         +--ro remote-discr?                discriminator
         +--ro new-state?                   state
         +--ro state-change-reason?         iana-bfd-types:diagnostic
         +--ro time-of-last-state-change?   yang:date-and-time
         +--ro dest-addr?                   inet:ip-address
         +--ro source-addr?                 inet:ip-address
         +--ro session-index?               uint32
         +--ro path-type?                   identityref
         +--ro interface?                   if:interface-ref
         +--ro echo-enabled?                boolean
```

2.7. BFD IP multihop hierarchy

   An "ip-mh" node is added under the "bfd" node in cntrol-plane-
   protocol. The configuration and operational state data for each BFD
   IP multihop session is under this "ip-mh" node. In the operational
   state model we support multiple BFD multihop sessions per remote
   address (ECMP), the local discriminator is used as key.

   module: ietf-bfd-ip-mh

```
      augment /rt:routing/rt:control-plane-protocols
              /rt:control-plane-protocol/bfd:bfd:
        +--rw ip-mh
           +--ro summary
           |  +--ro number-of-sessions?              yang:gauge32
           |  +--ro number-of-sessions-up?           yang:gauge32
           |  +--ro number-of-sessions-down?         yang:gauge32
           |  +--ro number-of-sessions-admin-down?   yang:gauge32
           +--rw session-groups
              +--rw session-group* [source-addr dest-addr]
                 +--rw source-addr                        inet:ip-address
                 +--rw dest-addr                          inet:ip-address
                 +--rw local-multiplier?                  multiplier
                 +--rw (interval-config-type)?
                 |  +--:(tx-rx-intervals)
                 |  |  +--rw desired-min-tx-interval?    uint32
                 |  |  +--rw required-min-rx-interval?   uint32
                 |  +--:(single-interval) {single-minimum-interval}?
                 |     +--rw min-interval?               uint32
                 +--rw demand-enabled?                    boolean
                 |       {demand-mode}?
                 +--rw admin-down?                        boolean
                 +--rw authentication! {authentication}?
                 |  +--rw key-chain?     kc:key-chain-ref
                 |  +--rw meticulous?    boolean
                 +--rw tx-ttl?                            bfd-types:hops
                 +--rw rx-ttl                             bfd-types:hops
                 +--ro sessions* []
                    +--ro path-type?             identityref
                    +--ro ip-encapsulation?      boolean
                    +--ro local-discriminator?   discriminator
                    +--ro remote-discriminator?  discriminator
                    +--ro remote-multiplier?     multiplier
                    +--ro demand-capability?     boolean {demand-mode}?
                    +--ro source-port?           inet:port-number
                    +--ro dest-port?             inet:port-number
                    +--ro session-running
                    |  +--ro session-index?                uint32
                    |  +--ro local-state?                  state
                    |  +--ro remote-state?                 state
                    |  +--ro local-diagnostic?
                    |  |      iana-bfd-types:diagnostic
                    |  +--ro remote-diagnostic?
                    |  |      iana-bfd-types:diagnostic
                    |  +--ro remote-authenticated?         boolean
                    |  +--ro remote-authentication-type?
                    |  |      iana-bfd-types:auth-type {authentication}?
                    |  +--ro detection-mode?               enumeration
```

```
                  | +--ro negotiated-tx-interval?       uint32
                  | +--ro negotiated-rx-interval?       uint32
                  | +--ro detection-time?               uint32
                  | +--ro echo-tx-interval-in-use?      uint32
                  |         {echo-mode}?
                  +--ro session-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 down-count?
                     |       yang:counter32
                     +--ro admin-down-count?
                     |       yang:counter32
                     +--ro receive-packet-count?
                     |       yang:counter64
                     +--ro send-packet-count?
                     |       yang:counter64
                     +--ro receive-invalid-packet-count?
                     |       yang:counter64
                     +--ro send-failed-packet-count?
                             yang:counter64

    notifications:
      +---n multihop-notification
         +--ro local-discr?                discriminator
         +--ro remote-discr?               discriminator
         +--ro new-state?                  state
         +--ro state-change-reason?        iana-bfd-types:diagnostic
         +--ro time-of-last-state-change?  yang:date-and-time
         +--ro dest-addr?                  inet:ip-address
         +--ro source-addr?                inet:ip-address
         +--ro session-index?              uint32
         +--ro path-type?                  identityref
```

2.8. BFD over LAG hierarchy

   A "lag" node is added under the "bfd" node in control-plane-protocol.
   The configuration and operational state data for each BFD LAG session
   is under this "lag" node.


```
   module: ietf-bfd-lag
     augment /rt:routing/rt:control-plane-protocols
             /rt:control-plane-protocol/bfd:bfd:
       +--rw lag
```

```
        +--rw micro-bfd-ipv4-session-statistics
        |  +--ro summary
        |     +--ro number-of-sessions?            yang:gauge32
        |     +--ro number-of-sessions-up?         yang:gauge32
        |     +--ro number-of-sessions-down?       yang:gauge32
        |     +--ro number-of-sessions-admin-down? yang:gauge32
        +--rw micro-bfd-ipv6-session-statistics
        |  +--ro summary
        |     +--ro number-of-sessions?            yang:gauge32
        |     +--ro number-of-sessions-up?         yang:gauge32
        |     +--ro number-of-sessions-down?       yang:gauge32
        |     +--ro number-of-sessions-admin-down? yang:gauge32
        +--rw sessions
           +--rw session* [lag-name]
              +--rw lag-name                     if:interface-ref
              +--rw ipv4-dest-addr?
              |      inet:ipv4-address
              +--rw ipv6-dest-addr?
              |      inet:ipv6-address
              +--rw local-multiplier?            multiplier
              +--rw (interval-config-type)?
              |  +--:(tx-rx-intervals)
              |  |  +--rw desired-min-tx-interval?   uint32
              |  |  +--rw required-min-rx-interval?  uint32
              |  +--:(single-interval) {single-minimum-interval}?
              |     +--rw min-interval?              uint32
              +--rw demand-enabled?              boolean
              |      {demand-mode}?
              +--rw admin-down?                  boolean
              +--rw authentication! {authentication}?
              |  +--rw key-chain?    kc:key-chain-ref
              |  +--rw meticulous?   boolean
              +--rw use-ipv4?                    boolean
              +--rw use-ipv6?                    boolean
              +--ro member-links* [member-link]
                 +--ro member-link      if:interface-ref
                 +--ro micro-bfd-ipv4
                 |  +--ro path-type?             identityref
                 |  +--ro ip-encapsulation?      boolean
                 |  +--ro local-discriminator?   discriminator
                 |  +--ro remote-discriminator?  discriminator
                 |  +--ro remote-multiplier?     multiplier
                 |  +--ro demand-capability?     boolean
                 |  |      {demand-mode}?
                 |  +--ro source-port?           inet:port-number
                 |  +--ro dest-port?             inet:port-number
                 |  +--ro session-running
                 |  |  +--ro session-index?              uint32
```

```
              │ │ +--ro local-state?                   state
              │ │ +--ro remote-state?                  state
              │ │ +--ro local-diagnostic?
              │ │ │       iana-bfd-types:diagnostic
              │ │ +--ro remote-diagnostic?
              │ │ │       iana-bfd-types:diagnostic
              │ │ +--ro remote-authenticated?          boolean
              │ │ +--ro remote-authentication-type?
              │ │ │       iana-bfd-types:auth-type
              │ │ │       {authentication}?
              │ │ +--ro detection-mode?                enumeration
              │ │ +--ro negotiated-tx-interval?        uint32
              │ │ +--ro negotiated-rx-interval?        uint32
              │ │ +--ro detection-time?                uint32
              │ │ +--ro echo-tx-interval-in-use?       uint32
              │ │         {echo-mode}?
              │ +--ro session-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 down-count?
              │    │       yang:counter32
              │    +--ro admin-down-count?
              │    │       yang:counter32
              │    +--ro receive-packet-count?
              │    │       yang:counter64
              │    +--ro send-packet-count?
              │    │       yang:counter64
              │    +--ro receive-invalid-packet-count?
              │    │       yang:counter64
              │    +--ro send-failed-packet-count?
              │            yang:counter64
              +--ro micro-bfd-ipv6
                 +--ro path-type?            identityref
                 +--ro ip-encapsulation?     boolean
                 +--ro local-discriminator?  discriminator
                 +--ro remote-discriminator? discriminator
                 +--ro remote-multiplier?    multiplier
                 +--ro demand-capability?    boolean
                 │       {demand-mode}?
                 +--ro source-port?          inet:port-number
                 +--ro dest-port?            inet:port-number
                 +--ro session-running
                 │  +--ro session-index?                uint32
                 │  +--ro local-state?                  state
```

```
                      | +--ro remote-state?                state
                      | +--ro local-diagnostic?
                      | |       iana-bfd-types:diagnostic
                      | +--ro remote-diagnostic?
                      | |       iana-bfd-types:diagnostic
                      | +--ro remote-authenticated?        boolean
                      | +--ro remote-authentication-type?
                      | |       iana-bfd-types:auth-type
                      | |       {authentication}?
                      | +--ro detection-mode?              enumeration
                      | +--ro negotiated-tx-interval?      uint32
                      | +--ro negotiated-rx-interval?      uint32
                      | +--ro detection-time?              uint32
                      | +--ro echo-tx-interval-in-use?     uint32
                      |       {echo-mode}?
                   +--ro session-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 down-count?
                      |       yang:counter32
                      +--ro admin-down-count?
                      |       yang:counter32
                      +--ro receive-packet-count?
                      |       yang:counter64
                      +--ro send-packet-count?
                      |       yang:counter64
                      +--ro receive-invalid-packet-count?
                      |       yang:counter64
                      +--ro send-failed-packet-count?
                              yang:counter64

  notifications:
    +---n lag-notification
       +--ro local-discr?                  discriminator
       +--ro remote-discr?                 discriminator
       +--ro new-state?                    state
       +--ro state-change-reason?          iana-bfd-types:diagnostic
       +--ro time-of-last-state-change?    yang:date-and-time
       +--ro dest-addr?                    inet:ip-address
       +--ro source-addr?                  inet:ip-address
       +--ro session-index?                uint32
       +--ro path-type?                    identityref
       +--ro lag-name?                     if:interface-ref
       +--ro member-link?                  if:interface-ref
```

2.9. BFD over MPLS LSPs hierarchy

   An "mpls" node is added under the "bfd" node in control-plane-
   protocol. The configuration is per MPLS FEC under this "mpls" node.
   In the operational state model we support multiple BFD sessions per
   MPLS FEC (ECMP), the local discriminator is used as key. The "mpls"
   node can be used in a network device (top-level), or mounted in an
   LNE or in a network instance.

```
   module: ietf-bfd-mpls
     augment /rt:routing/rt:control-plane-protocols
             /rt:control-plane-protocol/bfd:bfd:
       +--rw mpls
          +--ro summary
          |  +--ro number-of-sessions?             yang:gauge32
          |  +--ro number-of-sessions-up?          yang:gauge32
          |  +--ro number-of-sessions-down?        yang:gauge32
          |  +--ro number-of-sessions-admin-down?  yang:gauge32
          +--rw egress
          |  +--rw enable?                         boolean
          |  +--rw local-multiplier?               multiplier
          |  +--rw (interval-config-type)?
          |  |  +--:(tx-rx-intervals)
          |  |  |  +--rw desired-min-tx-interval?   uint32
          |  |  |  +--rw required-min-rx-interval?  uint32
          |  |  +--:(single-interval) {single-minimum-interval}?
          |  |     +--rw min-interval?              uint32
          |  +--rw authentication! {authentication}?
          |     +--rw key-chain?    kc:key-chain-ref
          |     +--rw meticulous?   boolean
          +--rw session-groups
             +--rw session-group* [mpls-fec]
                +--rw mpls-fec                      inet:ip-prefix
                +--rw local-multiplier?             multiplier
                +--rw (interval-config-type)?
                |  +--:(tx-rx-intervals)
                |  |  +--rw desired-min-tx-interval?   uint32
                |  |  +--rw required-min-rx-interval?  uint32
                |  +--:(single-interval) {single-minimum-interval}?
                |     +--rw min-interval?              uint32
                +--rw demand-enabled?               boolean
                |      {demand-mode}?
                +--rw admin-down?                   boolean
                +--rw authentication! {authentication}?
                |  +--rw key-chain?    kc:key-chain-ref
                |  +--rw meticulous?   boolean
                +--ro sessions* []
```

```
                 +--ro path-type?              identityref
                 +--ro ip-encapsulation?       boolean
                 +--ro local-discriminator?    discriminator
                 +--ro remote-discriminator?   discriminator
                 +--ro remote-multiplier?      multiplier
                 +--ro demand-capability?      boolean {demand-mode}?
                 +--ro source-port?            inet:port-number
                 +--ro dest-port?              inet:port-number
                 +--ro session-running
                 |  +--ro session-index?              uint32
                 |  +--ro local-state?                state
                 |  +--ro remote-state?               state
                 |  +--ro local-diagnostic?
                 |  |     iana-bfd-types:diagnostic
                 |  +--ro remote-diagnostic?
                 |  |     iana-bfd-types:diagnostic
                 |  +--ro remote-authenticated?       boolean
                 |  +--ro remote-authentication-type?
                 |  |     iana-bfd-types:auth-type {authentication}?
                 |  +--ro detection-mode?             enumeration
                 |  +--ro negotiated-tx-interval?     uint32
                 |  +--ro negotiated-rx-interval?     uint32
                 |  +--ro detection-time?             uint32
                 |  +--ro echo-tx-interval-in-use?    uint32
                 |        {echo-mode}?
                 +--ro session-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 down-count?
                 |  |     yang:counter32
                 |  +--ro admin-down-count?
                 |  |     yang:counter32
                 |  +--ro receive-packet-count?
                 |  |     yang:counter64
                 |  +--ro send-packet-count?
                 |  |     yang:counter64
                 |  +--ro receive-invalid-packet-count?
                 |  |     yang:counter64
                 |  +--ro send-failed-packet-count?
                 |        yang:counter64
                 +--ro mpls-dest-address?      inet:ip-address

      notifications:
        +---n mpls-notification
```

```
             +--ro local-discr?                  discriminator
             +--ro remote-discr?                 discriminator
             +--ro new-state?                    state
             +--ro state-change-reason?          iana-bfd-types:diagnostic
             +--ro time-of-last-state-change?    yang:date-and-time
             +--ro dest-addr?                    inet:ip-address
             +--ro source-addr?                  inet:ip-address
             +--ro session-index?                uint32
             +--ro path-type?                    identityref
             +--ro mpls-dest-address?            inet:ip-address
```

2.10. BFD over MPLS-TE hierarchy

   YANG Data Model for TE Topologies [I-D.ietf-teas-yang-te] is
   augmented. BFD is configured per MPLS-TE tunnel, and BFD session
   operational state data is provided per MPLS-TE LSP.


```
   module: ietf-bfd-mpls-te
     augment /rt:routing/rt:control-plane-protocols
             /rt:control-plane-protocol/bfd:bfd:
       +--rw mpls-te
          +--rw egress
          |  +--rw enable?                       boolean
          |  +--rw local-multiplier?             multiplier
          |  +--rw (interval-config-type)?
          |  |  +--:(tx-rx-intervals)
          |  |  |  +--rw desired-min-tx-interval?   uint32
          |  |  |  +--rw required-min-rx-interval?  uint32
          |  |  +--:(single-interval) {single-minimum-interval}?
          |  |     +--rw min-interval?              uint32
          |  +--rw authentication! {authentication}?
          |     +--rw key-chain?    kc:key-chain-ref
          |     +--rw meticulous?   boolean
          +--ro summary
             +--ro number-of-sessions?            yang:gauge32
             +--ro number-of-sessions-up?         yang:gauge32
             +--ro number-of-sessions-down?       yang:gauge32
             +--ro number-of-sessions-admin-down? yang:gauge32
     augment /te:te/te:tunnels/te:tunnel:
       +--rw local-multiplier?                 multiplier
       +--rw (interval-config-type)?
       |  +--:(tx-rx-intervals)
       |  |  +--rw desired-min-tx-interval?   uint32
       |  |  +--rw required-min-rx-interval?  uint32
       |  +--:(single-interval) {single-minimum-interval}?
       |     +--rw min-interval?              uint32
       +--rw demand-enabled?                   boolean {demand-mode}?
```

```
      +--rw admin-down?                         boolean
      +--rw authentication! {authentication}?
      |  +--rw key-chain?     kc:key-chain-ref
      |  +--rw meticulous?    boolean
      +--rw encap?                              identityref
   augment /te:te/te:lsps-state/te:lsp:
      +--ro path-type?              identityref
      +--ro ip-encapsulation?       boolean
      +--ro local-discriminator?    discriminator
      +--ro remote-discriminator?   discriminator
      +--ro remote-multiplier?      multiplier
      +--ro demand-capability?      boolean {demand-mode}?
      +--ro source-port?            inet:port-number
      +--ro dest-port?              inet:port-number
      +--ro session-running
      |  +--ro session-index?                 uint32
      |  +--ro local-state?                   state
      |  +--ro remote-state?                  state
      |  +--ro local-diagnostic?              iana-bfd-types:diagnostic
      |  +--ro remote-diagnostic?             iana-bfd-types:diagnostic
      |  +--ro remote-authenticated?          boolean
      |  +--ro remote-authentication-type?    iana-bfd-types:auth-type
      |  |       {authentication}?
      |  +--ro detection-mode?                enumeration
      |  +--ro negotiated-tx-interval?        uint32
      |  +--ro negotiated-rx-interval?        uint32
      |  +--ro detection-time?                uint32
      |  +--ro echo-tx-interval-in-use?       uint32 {echo-mode}?
      +--ro session-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 down-count?                     yang:counter32
      |  +--ro admin-down-count?               yang:counter32
      |  +--ro receive-packet-count?           yang:counter64
      |  +--ro send-packet-count?              yang:counter64
      |  +--ro receive-invalid-packet-count?   yang:counter64
      |  +--ro send-failed-packet-count?       yang:counter64
      +--ro mpls-dest-address?       inet:ip-address

   notifications:
      +---n mpls-te-notification
         +--ro local-discr?                discriminator
         +--ro remote-discr?               discriminator
         +--ro new-state?                  state
         +--ro state-change-reason?        iana-bfd-types:diagnostic
         +--ro time-of-last-state-change?  yang:date-and-time
         +--ro dest-addr?                  inet:ip-address
```

```
              +--ro source-addr?              inet:ip-address
              +--ro session-index?            uint32
              +--ro path-type?                identityref
              +--ro mpls-dest-address?        inet:ip-address
              +--ro tunnel-name?              string
```

2.11. Interaction with other YANG modules

   Generic YANG Data Model for Connectionless OAM protocols
   [I-D.ietf-lime-yang-connectionless-oam] describes how the LIME
   connectionless OAM model could be extended to support BFD.

   Also, the operation of the BFD data model depends on configuration
   parameters that are defined in other YANG modules.

2.11.1. Module ietf-interfaces

   The following boolean configuration is defined in A YANG Data Model
   for Interface Management [RFC8343]:

   /if:interfaces/if:interface/if:enabled
          If this configuration is set to "false", no BFD packets can
          be transmitted or received on that interface.

2.11.2. Module ietf-ip

   The following boolean configuration is defined in A YANG Data Model
   for IP Management [RFC8344]:

   /if:interfaces/if:interface/ip:ipv4/ip:enabled
          If this configuration is set to "false", no BFD IPv4 packets
          can be transmitted or received on that interface.

   /if:interfaces/if:interface/ip:ipv4/ip:forwarding
          If this configuration is set to "false", no BFD IPv4 packets
          can be transmitted or received on that interface.

   /if:interfaces/if:interface/ip:ipv6/ip:enabled
          If this configuration is set to "false", no BFD IPv6 packets
          can be transmitted or received on that interface.

   /if:interfaces/if:interface/ip:ipv6/ip:forwarding
          If this configuration is set to "false", no BFD IPv6 packets
          can be transmitted or received on that interface.

2.11.3. Module ietf-mpls

   The following boolean configuration is defined in A YANG Data Model
   for MPLS Base [I-D.ietf-mpls-base-yang]:

   /rt:routing/mpls:mpls/mpls:interface/mpls:config/mpls:enabled
           If this configuration is set to "false", no BFD MPLS packets
           can be transmitted or received on that interface.

2.11.4. Module ietf-te

   The following configuration is defined in the "ietf-te" YANG module
   YANG Data Model for TE Topology [I-D.ietf-teas-yang-te]:

   /ietf-te:te/ietf-te:tunnels/ietf-te:tunnel/ietf-te:config/ietf-
   te:admin-status
           If this configuration is not set to "state-up", no BFD MPLS
           packets can be transmitted or received on that tunnel.

2.12. IANA BFD YANG Module

   <CODE BEGINS> file "iana-bfd-types@2018-08-01.yang"

   module iana-bfd-types {

     yang-version 1.1;

     namespace "urn:ietf:params:xml:ns:yang:iana-bfd-types";

     prefix "iana-bfd-types";

     organization "IANA";

     contact
       "   Internet Assigned Numbers Authority

        Postal: ICANN
                12025 Waterfront Drive, Suite 300
                Los Angeles, CA 90094-2536
                United States of America

        Tel: +1 310 823 9358
        <mailto:iana@iana.org>";

     description
       "This module defines YANG data types for IANA-registered
        BFD parameters.

      This YANG module is maintained by IANA and reflects the
      'BFD Diagnostic Codes' and 'BFD Authentication Types' registries.

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

      Redistribution and use in source and binary forms, with or
      without modification, is permitted pursuant to, and subject
      to the license terms contained in, the Simplified BSD License
      set forth in Section 4.c of the IETF Trust's Legal Provisions
      Relating to IETF Documents
      (http://trustee.ietf.org/license-info).

      This version of this YANG module is part of RFC XXXX; see
      the RFC itself for full legal notices.";

   // RFC Ed.: replace XXXX with actual RFC number and remove
   // this note

   reference "RFC XXXX";

   revision 2018-08-01 {
     description "Initial revision.";
     reference "RFC XXXX: IANA BFD YANG Data Types.";
   }

   /*
    * Type Definitions
    */
   typedef diagnostic {
     type enumeration {
       enum none {
         value 0;
         description "None";
       }
       enum control-expiry {
         value 1;
         description "Control timer expiry";
       }
       enum echo-failed {
         value 2;
         description "Echo failure";
       }
       enum neighbor-down {
         value 3;
         description "Neighbor down";
       }
       enum forwarding-reset {

```
          value 4;
          description "Forwarding reset";
        }
        enum path-down {
          value 5;
          description "Path down";
        }
        enum concatenated-path-down {
          value 6;
          description "Concatenated path down";
        }
        enum admin-down {
          value 7;
          description "Admin down";
        }
        enum reverse-concatenated-path-down {
          value 8;
          description "Reverse concatenated path down";
        }
        enum mis-connectivity-defect {
          value 9;
          description "Mis-connectivity defect as specified in RFC6428";
        }
      }
      description
        "BFD diagnostic as defined in RFC 5880, values are maintained in
        the 'BFD Diagnostic Codes' IANA registry. Range is 0 to 31.";
    }

    typedef auth-type {
      type enumeration {
        enum reserved {
          value 0;
          description "Reserved";
        }
        enum simple-password {
          value 1;
          description "Simple password";
        }
        enum keyed-md5 {
          value 2;
          description "Keyed MD5";
        }
        enum meticulous-keyed-md5 {
          value 3;
          description "Meticulous keyed MD5";
        }
        enum keyed-sha1 {
```

```
          value 4;
          description "Keyed SHA1";
        }
        enum meticulous-keyed-sha1 {
          value 5;
          description "Meticulous keyed SHA1";
        }
      }
      description
        "BFD authentication type as defined in RFC 5880, values are
         maintained in the 'BFD Authentication Types' IANA registry.
         Range is 0 to 255.";
    }
  }

  <CODE ENDS>
```

2.13. BFD types YANG Module

   This YANG module imports typedefs from [RFC6991], [RFC8177] and the
   "control-plane-protocol" identity from [RFC8349].

```
<CODE BEGINS> file "ietf-bfd-types@2018-08-01.yang"

module ietf-bfd-types {

  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-bfd-types";

  prefix "bfd-types";

  // RFC Ed.: replace occurences of XXXX with actual RFC number and
  // remove this note

  import iana-bfd-types {
    prefix "iana-bfd-types";
    reference "RFC XXXX: YANG Data Model for BFD";
  }

  import ietf-inet-types {
    prefix "inet";
    reference "RFC 6991: Common YANG Data Types";
  }

  import ietf-yang-types {
    prefix "yang";
    reference "RFC 6991: Common YANG Data Types";
```

```
   }

   import ietf-routing {
     prefix "rt";
     reference
       "RFC 8349: A YANG Data Model for Routing Management
        (NMDA version)";
   }

   import ietf-key-chain {
     prefix "kc";
     reference "RFC 8177: YANG Data Model for Key Chains";
   }

   organization "IETF BFD Working Group";

   contact
     "WG Web:   <http://tools.ietf.org/wg/bfd>
      WG List:  <rtg-bfd@ietf.org>

      Editors:  Reshad Rahman (rrahman@cisco.com),
                Lianshu Zheng (vero.zheng@huawei.com),
                Mahesh Jethanandani (mjethanandani@gmail.com)";

   description
     "This module contains a collection of BFD specific YANG data type
      definitions, as per RFC 5880, and also groupings which are common
      to other BFD YANG modules.

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

      Redistribution and use in source and binary forms, with or
      without modification, is permitted pursuant to, and subject
      to the license terms contained in, the Simplified BSD License
      set forth in Section 4.c of the IETF Trust's Legal Provisions
      Relating to IETF Documents
      (http://trustee.ietf.org/license-info).

      This version of this YANG module is part of RFC XXXX; see
      the RFC itself for full legal notices.";

   reference "RFC XXXX";

   revision 2018-08-01 {
     description "Initial revision.";
     reference "RFC XXXX: YANG Data Model for BFD";
   }
```

```
   /*
    * Feature definitions
    */
   feature single-minimum-interval {
     description
       "This feature indicates that the server supports configuration
        of one minimum interval value which is used for both transmit and
        receive minimum intervals.";
   }

   feature authentication {
     description
       "This feature indicates that the server supports BFD
        authentication.";
     reference
       "RFC 5880: Bidirectional Forwarding Detection (BFD),
        section 6.7.";
   }

   feature demand-mode {
     description
       "This feature indicates that the server supports BFD demand
        mode.";
     reference
       "RFC 5880: Bidirectional Forwarding Detection (BFD),
        section 6.6.";
   }

   feature echo-mode {
     description
       "This feature indicates that the server supports BFD echo
        mode.";
     reference
       "RFC 5880: Bidirectional Forwarding Detection (BFD),
        section 6.4.";
   }

   /*
    * Identity definitions
    */
   identity bfdv1 {
     base "rt:control-plane-protocol";
     description "BFD protocol version 1.";
     reference
       "RFC 5880: Bidirectional Forwarding Detection (BFD).";
   }

   identity path-type {
```

```
     description
       "Base identity for BFD path type. The path type indicates
        the type of path on which BFD is running.";
   }
   identity path-ip-sh {
     base path-type;
     description "BFD on IP single hop.";
     reference
       "RFC 5881: Bidirectional Forwarding Detection (BFD)
        for IPv4 and IPv6 (Single Hop).";
   }
   identity path-ip-mh {
     base path-type;
     description "BFD on IP multihop paths.";
     reference
       "RFC 5883: Bidirectional Forwarding Detection (BFD) for
        Multihop Paths.";
   }
   identity path-mpls-te {
     base path-type;
     description
       "BFD on MPLS Traffic Engineering.";
     reference
       "RFC 5884: Bidirectional Forwarding Detection (BFD)
        for MPLS Label Switched Paths (LSPs).";
   }
   identity path-mpls-lsp {
     base path-type;
     description
       "BFD on MPLS Label Switched Path.";
     reference
       "RFC 5884: Bidirectional Forwarding Detection (BFD)
        for MPLS Label Switched Paths (LSPs).";
   }
   identity path-lag {
     base path-type;
     description
       "Micro-BFD on LAG member links.";
     reference
       "RFC 7130: Bidirectional Forwarding Detection (BFD) on
        Link Aggregation Group (LAG) Interfaces.";
   }

   identity encap-type {
     description
       "Base identity for BFD encapsulation type.";
   }
   identity encap-ip {
```

```
      base encap-type;
      description "BFD with IP encapsulation.";
    }

    /*
     * Type Definitions
     */
    typedef discriminator {
      type uint32;
      description "BFD discriminator as described in RFC 5880.";
    }

    typedef state {
      type enumeration {
        enum adminDown {
          value 0;
          description "admindown";
        }
        enum down {
          value 1;
          description "down";
        }
        enum init {
          value 2;
          description "init";
        }
        enum up {
          value 3;
          description "up";
        }
      }
      description "BFD state as defined in RFC 5880.";
    }

    typedef multiplier {
      type uint8 {
        range 1..255;
      }
      description "BFD multiplier as described in RFC 5880.";
    }

    typedef hops {
      type uint8 {
        range 1..255;
      }
      description
        "This corresponds to Time To Live for IPv4 and corresponds to hop
         limit for IPv6.";
```

```
   }

   /*
    * Groupings
    */
   grouping auth-parms {
     description
       "Grouping for BFD authentication parameters
        (see section 6.7 of RFC 5880).";
     container authentication {
       if-feature authentication;
       presence
         "Enables BFD authentication (see section 6.7 of RFC 5880).";
       description "Parameters for BFD authentication.";

       leaf key-chain {
         type kc:key-chain-ref;
         description "Name of the key-chain as per RFC 8177.";
       }

       leaf meticulous {
         type boolean;
         description
           "Enables meticulous mode as described in section 6.7 " +
           "of RFC 5880.";
       }
     }
   }

   grouping base-cfg-parms {
     description "BFD grouping for base config parameters.";
     leaf local-multiplier {
       type multiplier;
       default 3;
       description "Multiplier transmitted by local system.";
     }

     choice interval-config-type {
       description
         "Two interval values or one value used for both transmit and
          receive.";
       case tx-rx-intervals {
         leaf desired-min-tx-interval {
           type uint32;
           units microseconds;
           default 1000000;
           description
             "Desired minimum transmit interval of control packets.";
```

```
            }

            leaf required-min-rx-interval {
              type uint32;
              units microseconds;
              default 1000000;
              description
                "Required minimum receive interval of control packets.";
            }
          }
          case single-interval {
            if-feature single-minimum-interval;

            leaf min-interval {
              type uint32;
              units microseconds;
              default 1000000;
              description
                "Desired minimum transmit interval and required " +
                "minimum receive interval of control packets.";
            }
          }
        }
      }

  grouping client-cfg-parms {
    description
      "BFD grouping for configuration parameters
       used by clients of BFD, e.g. IGP or MPLS.";

    leaf enable {
      type boolean;
      default false;
      description
        "Indicates whether the BFD is enabled.";
    }
    uses base-cfg-parms;
  }

  grouping common-cfg-parms {
    description
      "BFD grouping for common configuration parameters.";

    uses base-cfg-parms;

    leaf demand-enabled {
      if-feature demand-mode;
      type boolean;
```

```
          default false;
          description
            "To enable demand mode.";
        }

      leaf admin-down {
        type boolean;
        default false;
        description
          "Is the BFD session administratively down.";
      }
      uses auth-parms;
    }

  grouping all-session {
    description "BFD session operational information";
    leaf path-type {
      type identityref {
        base path-type;
      }
      config "false";
      description
        "BFD path type, this indicates the path type that BFD is
        running on.";
    }
    leaf ip-encapsulation {
      type boolean;
      config "false";
      description "Whether BFD encapsulation uses IP.";
    }
    leaf local-discriminator {
      type discriminator;
      config "false";
      description "Local discriminator.";
    }
    leaf remote-discriminator {
      type discriminator;
      config "false";
      description "Remote discriminator.";
    }
    leaf remote-multiplier {
      type multiplier;
      config "false";
      description "Remote multiplier.";
    }
    leaf demand-capability {
      if-feature demand-mode;
      type boolean;
```

```
        config "false";
        description "Local demand mode capability.";
      }
      leaf source-port {
        when "../ip-encapsulation = 'true'" {
          description
            "Source port valid only when IP encapsulation is used.";
        }
        type inet:port-number;
        config "false";
        description "Source UDP port";
      }
      leaf dest-port {
        when "../ip-encapsulation = 'true'" {
          description
            "Destination port valid only when IP encapsulation is used.";
        }
        type inet:port-number;
        config "false";
        description "Destination UDP port.";
      }

      container session-running {
        config "false";
        description "BFD session running information.";
        leaf session-index {
          type uint32;
          description
            "An index used to uniquely identify BFD sessions.";
        }
        leaf local-state {
          type state;
          description "Local state.";
        }
        leaf remote-state {
          type state;
          description "Remote state.";
        }
        leaf local-diagnostic {
          type iana-bfd-types:diagnostic;
          description "Local diagnostic.";
        }
        leaf remote-diagnostic {
          type iana-bfd-types:diagnostic;
          description "Remote diagnostic.";
        }
        leaf remote-authenticated {
          type boolean;
```

```
          description
            "Indicates whether incoming BFD control packets are
            authenticated.";
        }
        leaf remote-authentication-type {
          when "../remote-authenticated = 'true'" {
            description
              "Only valid when incoming BFD control packets are
               authenticated.";
          }
          if-feature authentication;
          type iana-bfd-types:auth-type;
          description
            "Authentication type of incoming BFD control packets.";
        }
        leaf detection-mode {
          type enumeration {
            enum async-with-echo {
              value "1";
              description "Async with echo.";
            }
            enum async-without-echo {
              value "2";
              description "Async without echo.";
            }
            enum demand-with-echo {
              value "3";
              description "Demand with echo.";
            }
            enum demand-without-echo {
              value "4";
              description "Demand without echo.";
            }
          }
          description "Detection mode.";
        }
        leaf negotiated-tx-interval {
          type uint32;
          units microseconds;
          description "Negotiated transmit interval.";
        }
        leaf negotiated-rx-interval {
          type uint32;
          units microseconds;
          description "Negotiated receive interval.";
        }
        leaf detection-time {
          type uint32;
```

```
         units microseconds;
         description "Detection time.";
       }
       leaf echo-tx-interval-in-use {
         when "../../path-type = 'bfd-types:path-ip-sh'" {
           description
             "Echo is supported for IP single-hop only.";
         }
         if-feature echo-mode;
         type uint32;
         units microseconds;
         description "Echo transmit interval in use.";
       }
     }

     container session-statistics {
       config "false";
       description "BFD per-session statistics.";

       leaf create-time {
         type yang:date-and-time;
         description
           "Time and date when this session was created.";
       }
       leaf last-down-time {
         type yang:date-and-time;
         description
           "Time and date of last time this session went down.";
       }
       leaf last-up-time {
         type yang:date-and-time;
         description
           "Time and date of last time this session went up.";
       }
       leaf down-count {
         type yang:counter32;
         description
           "The number of times this session has transitioned in the
            down state.";
       }
       leaf admin-down-count {
         type yang:counter32;
         description
           "The number of times this session has transitioned in the
            admin-down state.";
       }
       leaf receive-packet-count {
         type yang:counter64;
```

```
         description
           "Count of received packets in this session. This includes
            valid and invalid received packets.";
       }
       leaf send-packet-count {
         type yang:counter64;
         description "Count of sent packets in this session.";
       }
       leaf receive-invalid-packet-count {
         type yang:counter64;
         description
           "Count of invalid received packets in this session.";
       }
       leaf send-failed-packet-count {
         type yang:counter64;
         description
           "Count of packets which failed to be sent in this session.";
       }
     }
   }

   grouping session-statistics-summary {
     description "Grouping for session statistics summary.";
     container summary {
       config false;
       description "BFD session statistics summary.";
       leaf number-of-sessions {
         type yang:gauge32;
         description "Number of BFD sessions.";
       }
       leaf number-of-sessions-up {
         type yang:gauge32;
         description
           "Number of BFD sessions currently in up state (as defined
            in RFC 5880).";
       }
       leaf number-of-sessions-down {
         type yang:gauge32;
         description
           "Number of BFD sessions currently in down or init state
            but not admin-down (as defined in RFC 5880).";
       }
       leaf number-of-sessions-admin-down {
         type yang:gauge32;
         description
           "Number of BFD sessions currently in admin-down state (as
            defined in RFC 5880).";
       }
```

```
      }
    }

    grouping notification-parms {
      description
        "This group describes common parameters that will be sent " +
        "as part of BFD notification.";

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

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

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

      leaf state-change-reason {
        type iana-bfd-types:diagnostic;
        description "BFD state change reason.";
      }

      leaf time-of-last-state-change {
        type yang:date-and-time;
        description
          "Calendar time of previous state change.";
      }

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

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

      leaf session-index {
        type uint32;
        description "An index used to uniquely identify BFD sessions.";
      }
```

```
      leaf path-type {
        type identityref {
          base path-type;
        }
        description "BFD path type.";
      }
    }
  }
}
```

<CODE ENDS>

2.14. BFD top-level YANG Module

   This YANG module imports and augments "/routing/control-plane-
   protocols/control-plane-protocol" from [RFC8349].

   <CODE BEGINS> file "ietf-bfd@2018-08-01.yang"

   module ietf-bfd {

     yang-version 1.1;

     namespace "urn:ietf:params:xml:ns:yang:ietf-bfd";

     prefix "bfd";

     // RFC Ed.: replace occurences of XXXX with actual RFC number and
     // remove this note

     import ietf-bfd-types {
       prefix "bfd-types";
       reference "RFC XXXX: YANG Data Model for BFD";
     }

     import ietf-routing {
       prefix "rt";
       reference
         "RFC 8349: A YANG Data Model for Routing Management
          (NMDA version)";
     }

     organization "IETF BFD Working Group";

     contact
       "WG Web: <http://tools.ietf.org/wg/bfd>
        WG List: <rtg-bfd@ietf.org>

        Editors: Reshad Rahman (rrahman@cisco.com),
```

```
                   Lianshu Zheng (vero.zheng@huawei.com),
                   Mahesh Jethanandani (mjethanandani@gmail.com)";

     description
       "This module contains the YANG definition for BFD parameters as
        per RFC 5880.

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

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

     reference "RFC XXXX";

     revision 2018-08-01 {
       description "Initial revision.";
       reference "RFC XXXX: YANG Data Model for BFD";
     }

     augment "/rt:routing/rt:control-plane-protocols/"
           + "rt:control-plane-protocol" {
       when "derived-from-or-self(rt:type, 'bfd-types:bfdv1')" {
         description
           "This augmentation is only valid for a control-plane protocol
            instance of BFD (type 'bfdv1').";
       }
       description "BFD augmentation.";

       container bfd {
         description "BFD top level container.";

         uses bfd-types:session-statistics-summary;
       }
     }
   }

   <CODE ENDS>
```

2.15. BFD IP single-hop YANG Module

   This YANG module imports "interface-ref" from [RFC8343], typedefs
   from [RFC6991] and augments "/routing/control-plane-protocols/
   control-plane-protocol" from [RFC8349].

   <CODE BEGINS> file "ietf-bfd-ip-sh@2018-08-01.yang"

   module ietf-bfd-ip-sh {

     yang-version 1.1;

     namespace "urn:ietf:params:xml:ns:yang:ietf-bfd-ip-sh";

     prefix "bfd-ip-sh";

     // RFC Ed.: replace occurences of XXXX with actual RFC number and
     // remove this note

     import ietf-bfd-types {
       prefix "bfd-types";
       reference "RFC XXXX: YANG Data Model for BFD";
     }

     import ietf-bfd {
       prefix "bfd";
       reference "RFC XXXX: YANG Data Model for BFD";
     }

     import ietf-interfaces {
       prefix "if";
       reference
         "RFC 8343: A YANG Data Model for Interface Management";
     }

     import ietf-inet-types {
       prefix "inet";
       reference "RFC 6991: Common YANG Data Types";
     }

     import ietf-routing {
       prefix "rt";
       reference
         "RFC 8349: A YANG Data Model for Routing Management
          (NMDA version)";
     }

     organization "IETF BFD Working Group";

```
      contact
        "WG Web:    <http://tools.ietf.org/wg/bfd>
         WG List:   <rtg-bfd@ietf.org>

         Editors:   Reshad Rahman (rrahman@cisco.com),
                    Lianshu Zheng (vero.zheng@huawei.com),
                    Mahesh Jethanandani (mjethanandani@gmail.com)";

      description
        "This module contains the YANG definition for BFD IP single-hop
         as per RFC 5881.

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

         Redistribution and use in source and binary forms, with or
         without modification, is permitted pursuant to, and subject
         to the license terms contained in, the Simplified BSD License
         set forth in Section 4.c of the IETF Trust's Legal Provisions
         Relating to IETF Documents
         (http://trustee.ietf.org/license-info).

         This version of this YANG module is part of RFC XXXX; see
         the RFC itself for full legal notices.";

      reference "RFC XXXX";

      revision 2018-08-01 {
        description "Initial revision.";
        reference "RFC XXXX: A YANG data model for BFD IP single-hop";
      }

      /*
       * Augments
       */
      augment "/rt:routing/rt:control-plane-protocols/"
            + "rt:control-plane-protocol/bfd:bfd" {
        description "BFD augmentation for IP single-hop";
        container ip-sh {
          description "BFD IP single-hop top level container";

          uses bfd-types:session-statistics-summary;

          container sessions {
            description
              "BFD IP single-hop sessions.";
            list session {
              key "interface dest-addr";
```

```
            description "List of IP single-hop sessions.";
            leaf interface {
              type if:interface-ref;
              description
                "Interface on which the BFD session is running.";
            }
            leaf dest-addr {
              type inet:ip-address;
              description "IP address of the peer.";
            }
            leaf source-addr {
              type inet:ip-address;
              description "Local IP address.";
            }

            uses bfd-types:common-cfg-parms;

            uses bfd-types:all-session;
          }
        }
        list interfaces {
          key "interface";
          description "List of interfaces.";
          leaf interface {
            type if:interface-ref;
            description
              "BFD information for this interface.";
          }

          uses bfd-types:auth-parms;
        }
      }
    }

    /*
     * Notifications
     */
    notification singlehop-notification {
      description
        "Notification for BFD single-hop session state change. An " +
        "implementation may rate-limit notifications, e.g. when a " +
        "session is continuously changing state.";

      uses bfd-types:notification-parms;

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

```
        }

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

    }

    <CODE ENDS>
```

2.16. BFD IP multihop YANG Module

   This YANG module imports typedefs from [RFC6991] and augments
   "/routing/control-plane-protocols/control-plane-protocol" from
   [RFC8349].

```
 <CODE BEGINS> file "ietf-bfd-ip-mh@2018-08-01.yang"

 module ietf-bfd-ip-mh {

   yang-version 1.1;

   namespace "urn:ietf:params:xml:ns:yang:ietf-bfd-ip-mh";

   prefix "bfd-ip-mh";

   // RFC Ed.: replace occurences of XXXX with actual RFC number and
   // remove this note

   import ietf-bfd-types {
     prefix "bfd-types";
     reference "RFC XXXX: YANG Data Model for BFD";
   }

   import ietf-bfd {
     prefix "bfd";
     reference "RFC XXXX: YANG Data Model for BFD";
   }

   import ietf-inet-types {
     prefix "inet";
     reference "RFC 6991: Common YANG Data Types";
   }

   import ietf-routing {
     prefix "rt";
```

```
      reference
        "RFC 8349: A YANG Data Model for Routing Management
         (NMDA version)";
    }

    organization "IETF BFD Working Group";

    contact
      "WG Web:   <http://tools.ietf.org/wg/bfd>
       WG List:  <rtg-bfd@ietf.org>

       Editors:  Reshad Rahman (rrahman@cisco.com),
                 Lianshu Zheng (vero.zheng@huawei.com),
                 Mahesh Jethanandani (mjethanandani@gmail.com)";

    description
      "This module contains the YANG definition for BFD IP multi-hop
       as per RFC 5883.

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

       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Simplified BSD License
       set forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (http://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC XXXX; see
       the RFC itself for full legal notices.";

    reference "RFC XXXX";

    revision 2018-08-01 {
      description "Initial revision.";
      reference "RFC XXXX: A YANG data model for BFD IP multihop.";
    }

    /*
     * Augments
     */
    augment "/rt:routing/rt:control-plane-protocols/"
          + "rt:control-plane-protocol/bfd:bfd" {
      description "BFD augmentation for IP multihop.";
      container ip-mh {
        description "BFD IP multihop top level container.";
```

```
        uses bfd-types:session-statistics-summary;

        container session-groups {
          description
            "BFD IP multi-hop session groups.";
          list session-group {
            key "source-addr dest-addr";
            description
              "Group of BFD IP multi-hop sessions (for ECMP). A " +
              "group of sessions is between 1 source and 1  " +
              "destination, each session has a different field " +
              "in UDP/IP hdr for ECMP.";

            leaf source-addr {
              type inet:ip-address;
              description
                "Local IP address.";
            }
            leaf dest-addr {
              type inet:ip-address;
              description
                "IP address of the peer.";
            }
            uses bfd-types:common-cfg-parms;

            leaf tx-ttl {
              type bfd-types:hops;
              default 255;
              description "Hop count of outgoing BFD control packets.";
            }
            leaf rx-ttl {
              type bfd-types:hops;
              mandatory true;
              description
                "Minimum allowed hop count value for incoming BFD control
                 packets. Control packets whose hop count is lower than
                 this value are dropped.";
            }
            list sessions {
              config false;
              description
                "The multiple BFD sessions between a source and a " +
                "destination.";
              uses bfd-types:all-session;
            }
          }
        }
      }
```

```
    }

    /*
     * Notifications
     */
    notification multihop-notification {
      description
        "Notification for BFD multi-hop session state change. An " +
        "implementation may rate-limit notifications, e.g. when a " +
        "session is continuously changing state.";

      uses bfd-types:notification-parms;
    }
  }

 <CODE ENDS>
```

2.17. BFD over LAG YANG Module

   This YANG module imports "interface-ref" from [RFC8343], typedefs
   from [RFC6991] and augments "/routing/control-plane-protocols/
   control-plane-protocol" from [RFC8349].

```
  <CODE BEGINS> file "ietf-bfd-lag@2018-08-01.yang"

  module ietf-bfd-lag {

    yang-version 1.1;

    namespace "urn:ietf:params:xml:ns:yang:ietf-bfd-lag";

    prefix "bfd-lag";

    // RFC Ed.: replace occurences of XXXX with actual RFC number and
    // remove this note

    import ietf-bfd-types {
      prefix "bfd-types";
      reference "RFC XXXX: YANG Data Model for BFD";
    }

    import ietf-bfd {
      prefix "bfd";
      reference "RFC XXXX: YANG Data Model for BFD";
    }

    import ietf-interfaces {
      prefix "if";
```

```
      reference
        "RFC 8343: A YANG Data Model for Interface Management";
    }

    import ietf-inet-types {
      prefix "inet";
      reference "RFC 6991: Common YANG Data Types";
    }

    import ietf-routing {
      prefix "rt";
      reference
        "RFC 8349: A YANG Data Model for Routing Management
         (NMDA version)";
    }

    organization "IETF BFD Working Group";

    contact
      "WG Web:   <http://tools.ietf.org/wg/bfd>
       WG List:  <rtg-bfd@ietf.org>

       Editors:  Reshad Rahman (rrahman@cisco.com),
                 Lianshu Zheng vero.zheng@huawei.com),
                 Mahesh Jethanandani (mjethanandani@gmail.com)";

    description
      "This module contains the YANG definition for BFD over LAG
       interfaces as per RFC7130.

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

       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Simplified BSD License
       set forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (http://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC XXXX; see
       the RFC itself for full legal notices.";

    reference "RFC XXXX";

    revision 2018-08-01 {
      description "Initial revision.";
      reference "RFC XXXX: A YANG data model for BFD over LAG";
```

```
      }

      /*
       * Augments
       */
      augment "/rt:routing/rt:control-plane-protocols/"
            + "rt:control-plane-protocol/bfd:bfd" {
        description "BFD augmentation for LAG";
        container lag {
          description "BFD over LAG top level container";

          container micro-bfd-ipv4-session-statistics {
            description "Micro-BFD IPv4 session counters.";
            uses bfd-types:session-statistics-summary;
          }
          container micro-bfd-ipv6-session-statistics {
            description "Micro-BFD IPv6 session counters.";
            uses bfd-types:session-statistics-summary;
          }

          container sessions {
            description
              "BFD over LAG sessions";
            list session {
              key "lag-name";
              description "List of BFD over LAG sessions.";
              leaf lag-name {
                type if:interface-ref ;
                description "Name of the LAG";
              }
              leaf ipv4-dest-addr {
                type inet:ipv4-address;
                description
                  "IPv4 address of the peer, for IPv4 micro-BFD.";
              }
              leaf ipv6-dest-addr {
                type inet:ipv6-address;
                description
                  "IPv6 address of the peer, for IPv6 micro-BFD.";
              }
              uses bfd-types:common-cfg-parms;

              leaf use-ipv4 {
                type boolean;
                description "Using IPv4 micro-BFD.";
              }
              leaf use-ipv6 {
                type boolean;
```

```
                  description "Using IPv6 micro-BFD.";
               }

            list member-links {
               key "member-link";
               config false;
               description
                 "Micro-BFD over LAG. This represents one member link.";

               leaf member-link {
                 type if:interface-ref;
                 description
                   "Member link on which micro-BFD is running.";
               }
               container micro-bfd-ipv4 {
                 when "../../use-ipv4 = 'true'" {
                   description "Needed only if IPv4 is used.";
                 }
                 description
                   "Micro-BFD IPv4 session state on member link.";
                 uses bfd-types:all-session;
               }
               container micro-bfd-ipv6 {
                 when "../../use-ipv6 = 'true'" {
                   description "Needed only if IPv6 is used.";
                 }
                 description
                   "Micro-BFD IPv6 session state on member link.";
                 uses bfd-types:all-session;
               }
             }
           }
         }
       }

       /*
        * Notifications
        */
     notification lag-notification {
       description
         "Notification for BFD over LAG session state change. " +
         "An implementation may rate-limit notifications, e.g. when a " +
         "session is continuously changing state.";

       uses bfd-types:notification-parms;

       leaf lag-name {
```

```
          type if:interface-ref;
          description "LAG interface name.";
        }

        leaf member-link {
          type if:interface-ref;
          description "Member link on which BFD is running.";
        }
      }
    }
  }

  <CODE ENDS>
```

2.18. BFD over MPLS YANG Module

   This YANG module imports typedefs from [RFC6991] and augments
   "/routing/control-plane-protocols/control-plane-protocol" from
   [RFC8349].

   <CODE BEGINS> file "ietf-bfd-mpls@2018-08-01.yang"

```
  module ietf-bfd-mpls {

    yang-version 1.1;

    namespace "urn:ietf:params:xml:ns:yang:ietf-bfd-mpls";

    prefix "bfd-mpls";

    // RFC Ed.: replace occurences of XXXX with actual RFC number and
    // remove this note

    import ietf-bfd-types {
      prefix "bfd-types";
      reference "RFC XXXX: YANG Data Model for BFD";
    }

    import ietf-bfd {
      prefix "bfd";
      reference "RFC XXXX: YANG Data Model for BFD";
    }

    import ietf-inet-types {
      prefix "inet";
      reference "RFC 6991: Common YANG Data Types";
    }

    import ietf-routing {
```

```
      prefix "rt";
      reference
        "RFC 8349: A YANG Data Model for Routing Management
         (NMDA version)";
    }

    organization "IETF BFD Working Group";

    contact
      "WG Web:   <http://tools.ietf.org/wg/bfd>
       WG List:  <rtg-bfd@ietf.org>

       Editors:   Reshad Rahman (rrahman@cisco.com),
                  Lianshu Zheng (vero.zheng@huawei.com),
                  Mahesh Jethanandani (mjethanandani@gmail.com)";

    description
      "This module contains the YANG definition for BFD parameters for
       MPLS LSPs as per RFC 5884.

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

       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Simplified BSD License
       set forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (http://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC XXXX; see
       the RFC itself for full legal notices.";

    reference "RFC XXXX";

    revision 2018-08-01 {
      description "Initial revision.";
      reference "RFC XXXX: A YANG data model for BFD over MPLS LSPs";
    }

    /*
     * Identity definitions
     */
    identity encap-gach {
      base bfd-types:encap-type;
      description
        "BFD with G-ACh encapsulation as per RFC 5586.";
    }
```

```
    identity encap-ip-gach {
      base bfd-types:encap-type;
      description
        "BFD with IP and G-ACh encapsulation as per RFC 5586.";
    }

    /*
     * Groupings
     */
    grouping encap-cfg {
      description "Configuration for BFD encapsulation";

      leaf encap {
        type identityref {
          base bfd-types:encap-type;
        }
        default bfd-types:encap-ip;
        description "BFD encapsulation";
      }
    }

    grouping mpls-dest-address {
      description "Destination address as per RFC 5884.";

      leaf mpls-dest-address {
        type inet:ip-address;
        config "false";
        description
          "Destination address as per RFC 5884.
           Needed if IP encapsulation is used.";
      }
    }

    /*
     * Augments
     */
    augment "/rt:routing/rt:control-plane-protocols/"
          + "rt:control-plane-protocol/bfd:bfd" {
      description "BFD augmentation for MPLS.";
      container mpls {
        description "BFD MPLS top level container.";

        uses bfd-types:session-statistics-summary;

        container egress {
          description "Egress configuration.";

          uses bfd-types:client-cfg-parms;
```

```
            uses bfd-types:auth-parms;
          }

          container session-groups {
            description
              "BFD over MPLS session groups.";
            list session-group {
              key "mpls-fec";
              description
                "Group of BFD MPLS sessions (for ECMP). A group of " +
                "sessions is for 1 FEC, each session has a different " +
                "field in UDP/IP hdr for ECMP.";
              leaf mpls-fec {
                type inet:ip-prefix;
                description "MPLS FEC.";
              }

              uses bfd-types:common-cfg-parms;

              list sessions {
                config false;
                description
                  "The BFD sessions for an MPLS FEC. Local " +
                  "discriminator is unique for each session in the " +
                  "group.";
                uses bfd-types:all-session;

                uses bfd-mpls:mpls-dest-address;
              }
            }
          }
        }
      }

      /*
       * Notifications
       */
      notification mpls-notification {
        description
          "Notification for BFD over MPLS FEC session state change. " +
          "An implementation may rate-limit notifications, e.g. when a " +
          "session is continuously changing state.";

        uses bfd-types:notification-parms;

        leaf mpls-dest-address {
          type inet:ip-address;
          description
```

```
              "Destination address as per RFC 5884.
               Needed if IP encapsulation is used.";
        }
      }
   }

   <CODE ENDS>
```

2.19. BFD over MPLS-TE YANG Module

   This YANG module imports and augments "/te/tunnels/tunnel" from
   [I-D.ietf-teas-yang-te].

   <CODE BEGINS> file "ietf-bfd-mpls-te@2018-08-01.yang"

   module ietf-bfd-mpls-te {

     yang-version 1.1;

     namespace "urn:ietf:params:xml:ns:yang:ietf-bfd-mpls-te";

     prefix "bfd-mpls-te";

     // RFC Ed.: replace occurences of XXXX with actual RFC number and
     // remove this note

     import ietf-bfd-types {
       prefix "bfd-types";
       reference "RFC XXXX: YANG Data Model for BFD";
     }

     import ietf-bfd {
       prefix "bfd";
       reference "RFC XXXX: YANG Data Model for BFD";
     }

     import ietf-bfd-mpls {
       prefix "bfd-mpls";
       reference "RFC XXXX: YANG Data Model for BFD";
     }

     import ietf-te {
       prefix "te";
       // RFC Ed.: replace YYYY with actual RFC number of
       // draft-ietf-teas-yang-te and remove this note.
       reference
         "RFC YYYY: A YANG Data Model for Traffic Engineering Tunnels and
          Interfaces";
```

```
      }

      import ietf-routing {
        prefix "rt";
        reference
          "RFC 8349: A YANG Data Model for Routing Management
           (NMDA version)";
      }

      organization "IETF BFD Working Group";

      contact
        "WG Web:   <http://tools.ietf.org/wg/bfd>
         WG List:  <rtg-bfd@ietf.org>

         Editors:  Reshad Rahman (rrahman@cisco.com),
                   Lianshu Zheng (vero.zheng@huawei.com),
                   Mahesh Jethanandani (mjethanandani@gmail.com)";

      description
        "This module contains the YANG definition for BFD parameters for
         MPLS Traffic Engineering as per RFC 5884.

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

         Redistribution and use in source and binary forms, with or
         without modification, is permitted pursuant to, and subject
         to the license terms contained in, the Simplified BSD License
         set forth in Section 4.c of the IETF Trust's Legal Provisions
         Relating to IETF Documents
         (http://trustee.ietf.org/license-info).

         This version of this YANG module is part of RFC XXXX; see
         the RFC itself for full legal notices.";

      reference "RFC XXXX";

      revision 2018-08-01 {
        description "Initial revision.";
        reference "RFC XXXX: A YANG data model for BFD over MPLS-TE";
      }

      /*
       * Augments
       */
      augment "/rt:routing/rt:control-plane-protocols/"
            + "rt:control-plane-protocol/bfd:bfd" {
```

```
     description "BFD augmentation for MPLS-TE.";
     container mpls-te {
       description "BFD MPLS-TE top level container.";

       container egress {
         description "Egress configuration.";

         uses bfd-types:client-cfg-parms;

         uses bfd-types:auth-parms;
       }

       uses bfd-types:session-statistics-summary;
     }
   }

   augment "/te:te/te:tunnels/te:tunnel" {
     description "BFD configuration on MPLS-TE tunnel.";

     uses bfd-types:common-cfg-parms;

     uses bfd-mpls:encap-cfg;
   }

   augment "/te:te/te:lsps-state/te:lsp" {
     when "/te:te/te:lsps-state/te:lsp/te:origin-type != 'transit'" {
       description "BFD information not needed at transit points.";
     }
     description "BFD state information on MPLS-TE LSP.";

     uses bfd-types:all-session;

     uses bfd-mpls:mpls-dest-address;
   }

   /*
    * Notifications
    */
   notification mpls-te-notification {
     description
       "Notification for BFD over MPLS-TE session state change. " +
       "An implementation may rate-limit notifications, e.g. when a " +
       "session is continuously changing state.";

     uses bfd-types:notification-parms;

     uses bfd-mpls:mpls-dest-address;
```

```
      leaf tunnel-name {
        type string;
        description "MPLS-TE tunnel on which BFD was running.";
      }
    }
  }
```

   <CODE ENDS>

3. Data Model examples

   This section presents some simple and illustrative examples on how to
   configure BFD.

3.1. IP single-hop

   The following is an example configuration for a BFD IP single-hop
   session. The desired transmit interval and the required receive
   interval are both set to 10ms.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
    <interface>
      <name>eth0</name>
      <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
        ianaift:ethernetCsmacd
      </type>
    </interface>
  </interfaces>
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type xmlns:bfd-types=
            "urn:ietf:params:xml:ns:yang:ietf-bfd-types">
          bfd-types:bfdv1
        </type>
        <name>name:BFD</name>
        <bfd xmlns="urn:ietf:params:xml:ns:yang:ietf-bfd">
          <ip-sh xmlns="urn:ietf:params:xml:ns:yang:ietf-bfd-ip-sh">
            <sessions>
              <session>
                <interface>eth0</interface>
                <dest-addr>2001:db8:0:113::101</dest-addr>
                <desired-min-tx-interval>10000</desired-min-tx-interval>
                <required-min-rx-interval>
                  10000
                </required-min-rx-interval>
              </session>
            </sessions>
          </ip-sh>
        </bfd>
      </control-plane-protocol>
    </control-plane-protocols>
  </routing>
</config>
```

3.2. IP multihop

   The following is an example configuration for a BFD IP multihop
   session group. The desired transmit interval and the required
   receive interval are both set to 150ms.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type xmlns:bfd-types=
            "urn:ietf:params:xml:ns:yang:ietf-bfd-types">
          bfd-types:bfdv1
        </type>
        <name>name:BFD</name>
        <bfd xmlns="urn:ietf:params:xml:ns:yang:ietf-bfd">
          <ip-mh xmlns="urn:ietf:params:xml:ns:yang:ietf-bfd-ip-mh">
            <session-groups>
              <session-group>
                <source-addr>2001:db8:0:113::103</source-addr>
                <dest-addr>2001:db8:0:114::100</dest-addr>
                <desired-min-tx-interval>
                  150000
                </desired-min-tx-interval>
                <required-min-rx-interval>
                  150000
                </required-min-rx-interval>
                <rx-ttl>240</rx-ttl>
              </session-group>
            </session-groups>
          </ip-mh>
        </bfd>
      </control-plane-protocol>
    </control-plane-protocols>
  </routing>
</config>
```

3.3. LAG

   The following is an example of BFD configuration for a LAG session.
   In this case, an interface named "Bundle-Ether1" of interface type
   "ieee802eadLag" has a desired transmit and required receive interval
   set to 10ms.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
    <interface>
      <name>Bundle-Ether1</name>
      <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
        ianaift:ieee8023adLag
      </type>
    </interface>
  </interfaces>
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type xmlns:bfd-types=
            "urn:ietf:params:xml:ns:yang:ietf-bfd-types">
          bfd-types:bfdv1
        </type>
        <name>name:BFD</name>
        <bfd xmlns="urn:ietf:params:xml:ns:yang:ietf-bfd">
          <lag xmlns="urn:ietf:params:xml:ns:yang:ietf-bfd-lag">
            <sessions>
              <session>
                <lag-name>Bundle-Ether1</lag-name>
                <ipv6-dest-addr>2001:db8:112::16</ipv6-dest-addr>
                <desired-min-tx-interval>
                  100000
                </desired-min-tx-interval>
                <required-min-rx-interval>
                  100000
                </required-min-rx-interval>
                <use-ipv6>true</use-ipv6>
              </session>
            </sessions>
          </lag>
        </bfd>
      </control-plane-protocol>
    </control-plane-protocols>
  </routing>
</config>
```

3.4. MPLS

   The following is an example of BFD configured for an MPLS LSP. In
   this case, the desired transmit and required receive interval set to
   250ms.

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <routing xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type xmlns:bfd-types=
            "urn:ietf:params:xml:ns:yang:ietf-bfd-types">
          bfd-types:bfdv1
        </type>
        <name>name:BFD</name>
        <bfd xmlns="urn:ietf:params:xml:ns:yang:ietf-bfd">
          <mpls xmlns="urn:ietf:params:xml:ns:yang:ietf-bfd-mpls">
            <session-groups>
              <session-group>
                <mpls-fec>2001:db8:114::/116</mpls-fec>
                <desired-min-tx-interval>
                  250000
                </desired-min-tx-interval>
                <required-min-rx-interval>
                  250000
                </required-min-rx-interval>
              </session-group>
            </session-groups>
          </mpls>
        </bfd>
      </control-plane-protocol>
    </control-plane-protocols>
  </routing>
</config>
```

4. Security Considerations

   The YANG module specified in this document defines a schema for data
   that is designed to be accessed via network management protocols such
   as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer
   is the secure transport layer, and the mandatory-to-implement secure
   transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer
   is HTTPS, and the mandatory-to-implement secure transport is TLS
   [RFC5246].

   The NETCONF access control model [RFC6536] provides the means to
   restrict access for particular NETCONF or RESTCONF users to a
   preconfigured subset of all available NETCONF or RESTCONF protocol
   operations and content.

   There are a number of data nodes defined in this YANG module that are
   writable/creatable/deletable (i.e., config true, which is the

default). These data nodes may be considered sensitive or vulnerable
in some network environments. Write operations (e.g., edit-config)
to these data nodes without proper protection can have a negative
effect on network operations. These are the subtrees and data nodes
and their sensitivity/vulnerability:

/routing/control-plane-protocols/control-plane-protocol/bfd/ip-sh/
sessions: the list specifies the IP single-hop BFD sessions.

/routing/control-plane-protocols/control-plane-protocol/bfd/ip-sh/
sessions: data nodes local-multiplier, desired-min-tx-interval,
required-min-rx-interval and min-interval all impact the BFD IP
single-hop session. The source-addr and dest-addr data nodes can be
used to send BFD packets to unwitting recipients, [RFC5880] describes
how BFD mitigates against such threats. Authentication data nodes
key-chain and meticulous impact the security of the BFD IP single-hop
session.

/routing/control-plane-protocols/control-plane-protocol/bfd/ip-mh/
session-group: the list specifies the IP multi-hop BFD session
groups.

/routing/control-plane-protocols/control-plane-protocol/bfd/ip-mh/
session-group: data nodes local-multiplier, desired-min-tx-interval,
required-min-rx-interval and min-interval all impact the BFD IP
multi-hop session. The source-addr and dest-addr data nodes can be
used to send BFD packets to unwitting recipients, [RFC5880] describes
how BFD mitigates against such threats. Authentication data nodes
key-chain and meticulous impact the security of the BFD IP multi-hop
session.

/routing/control-plane-protocols/control-plane-protocol/bfd/lag/
sessions: the list specifies the BFD sessions over LAG.

/routing/control-plane-protocols/control-plane-protocol/bfd/lag/
sessions: data nodes local-multiplier, desired-min-tx-interval,
required-min-rx-interval and min-interval all impact the BFD over LAG
session. The ipv4-dest-addr and ipv6-dest-addr data nodes can be
used to send BFD packets to unwitting recipients, [RFC5880] describes
how BFD mitigates against such threats. Authentication data nodes
key-chain and meticulous impact the security of the BFD over LAG
session.

/routing/control-plane-protocols/control-plane-protocol/bfd/mpls/
session-group: the list specifies the session groups for BFD over
MPLS.

/routing/control-plane-protocols/control-plane-protocol/bfd/mpls/
session-group: data nodes local-multiplier, desired-min-tx-interval,
required-min-rx-interval, and min-interval all impact the BFD over
MPLS LSPs session. Authentication data nodes key-chain and
meticulous impact the security of the BFD over MPLS LSPs session.

/routing/control-plane-protocols/control-plane-protocol/bfd/mpls/
egress: data nodes local-multiplier, desired-min-tx-interval,
required-min-rx-interval and min-interval all impact the BFD over
MPLS LSPs sessions for which this device is an MPLS LSP egress node.
Authentication data nodes key-chain and meticulous impact the
security of the BFD over MPLS LSPs sessions for which this device is
an MPLS LSP egress node

/te/tunnels/tunnel: data nodes local-multiplier, desired-min-tx-
interval, required-min-rx-interval and min-interval all impact the
BFD session over the MPLS-TE tunnel. Authentication data nodes key-
chain and meticulous impact the security of the BFD session over the
MPLS-TE tunnel.

/routing/control-plane-protocols/control-plane-protocol/bfd/mpls-te/
egress: data nodes local-multiplier, desired-min-tx-interval,
required-min-rx-interval and min-interval all impact the BFD over
MPLS-TE sessions for which this device is an MPLS-TE egress node.
Authentication data nodes key-chain and meticulous impact the
security of the BFD over MPLS-TE sessions for which this device is an
MPLS-TE egress node.

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.

When BFD clients are used to modify BFD configuration (as described
in Section 2.1), the BFD clients need to be included in an analysis
of the security properties of the BFD-using system (e.g., when
considering the authentication and authorization of control actions).
In many cases, BFD is not the most vulnerable portion of such a
composite system, since BFD is limited to generating well-defined
traffic at a fixed rate on a given path; in the case of an IGP as BFD
client, attacking the IGP could cause more broad-scale disruption
than (de)configuring a BFD session could cause.

Some of the readable data nodes in this YANG module may be considered
sensitive or vulnerable in some network environments. It is thus
important to control read access (e.g., via get, get-config, or

notification) to these data nodes. These are the subtrees and data
nodes and their sensitivity/vulnerability:

/routing/control-plane-protocols/control-plane-protocol/bfd/ip-sh/
summary: access to this information discloses the number of BFD IP
single-hop sessions which are up, down and admin-down. The counters
include BFD sessions for which the user does not have read-access.

/routing/control-plane-protocols/control-plane-protocol/bfd/ip-
sh/sessions/session/: access to data nodes local-discriminator and
remote-discriminator (combined with the data nodes in the
authentication container) provides the ability to spoof BFD IP
single-hop packets.

/routing/control-plane-protocols/control-plane-protocol/bfd/ip-mh/
summary: access to this information discloses the number of BFD IP
multi-hop sessions which are up, down and admin-down. The counters
include BFD sessions for which the user does not have read-access.

/routing/control-plane-protocols/control-plane-protocol/bfd/ip-mh/
session-groups/session-group/sessions: access to data nodes local-
discriminator and remote-discriminator (combined with the data nodes
in the session-group's authentication container) provides the ability
to spoof BFD IP multi-hop packets.

/routing/control-plane-protocols/control-plane-protocol/bfd/lag/
micro-bfd-ipv4-session-statistics/summary: access to this information
discloses the number of micro BFD IPv4 LAG sessions which are up,
down and admin-down. The counters include BFD sessions for which the
user does not have read-access.

/routing/control-plane-protocols/control-plane-
protocol/bfd/lag/sessions/session/member-links/member-link/micro-bfd-
ipv4: access to data nodes local-discriminator and remote-
discriminator (combined with the data nodes in the session's
authentication container) provides the ability to spoof BFD IPv4 LAG
packets.

/routing/control-plane-protocols/control-plane-protocol/bfd/lag/
micro-bfd-ipv6-session-statistics/summary: access to this information
discloses the number of micro BFD IPv6 LAG sessions which are up,
down and admin-down. The counters include BFD sessions for which the
user does not have read-access.

/routing/control-plane-protocols/control-plane-
protocol/bfd/lag/sessions/session/member-links/member-link/micro-bfd-
ipv6: access to data nodes local-discriminator and remote-
discriminator (combined with the data nodes in the session's

authentication container) provides the ability to spoof BFD IPv6 LAG
packets.

/routing/control-plane-protocols/control-plane-protocol/bfd/mpls/
summary: access to this information discloses the number of BFD
sessions over MPLS LSPs which are up, down and admin-down. The
counters include BFD sessions for which the user does not have read-
access.

/routing/control-plane-protocols/control-plane-protocol/bfd/mpls/
session-groups/session-group/sessions: access to data nodes local-
discriminator and remote-discriminator (combined with the data nodes
in the session-group's authentication container) provides the ability
to spoof BFD over MPLS LSPs packets.

/routing/control-plane-protocols/control-plane-protocol/bfd/mpls-te/
summary: access to this information discloses the number of BFD
sessions over MPLS-TE which are up, down and admin-down. The
counters include BFD sessions for which the user does not have read-
access.

/te/lsps-state/lsp: access to data nodes local-discriminator and
remote-discriminator (combined with the data nodes in the tunnel's
authentication container) provides the ability to spoof BFD over
MPLS-TE packets.

5. IANA Considerations

   This document registers the following namespace URIs in the IETF XML
   registry [RFC3688]:

   URI: urn:ietf:params:xml:ns:yang:iana-bfd-types

   Registrant Contact: The IESG.

   XML: N/A, the requested URI is an XML namespace.

   URI: urn:ietf:params:xml:ns:yang:ietf-bfd-types

   Registrant Contact: The IESG.

   XML: N/A, the requested URI is an XML namespace.

```
   ---------------------------------------------------------------------

   ---------------------------------------------------------------------

   URI: urn:ietf:params:xml:ns:yang:ietf-bfd

   Registrant Contact: The IESG.

   XML: N/A, the requested URI is an XML namespace.

   ---------------------------------------------------------------------

   ---------------------------------------------------------------------

   URI: urn:ietf:params:xml:ns:yang:ietf-bfd-ip-sh

   Registrant Contact: The IESG.

   XML: N/A, the requested URI is an XML namespace.

   ---------------------------------------------------------------------

   ---------------------------------------------------------------------

   URI: urn:ietf:params:xml:ns:yang:ietf-bfd-mh

   Registrant Contact: The IESG.

   XML: N/A, the requested URI is an XML namespace.

   ---------------------------------------------------------------------

   ---------------------------------------------------------------------

   URI: urn:ietf:params:xml:ns:yang:ietf-bfd-lag

   Registrant Contact: The IESG.

   XML: N/A, the requested URI is an XML namespace.

   ---------------------------------------------------------------------

   ---------------------------------------------------------------------

   URI: urn:ietf:params:xml:ns:yang:ietf-bfd-mpls

   Registrant Contact: The IESG.
```

XML: N/A, the requested URI is an XML namespace.

```
----------------------------------------------------------------

----------------------------------------------------------------
```

URI: urn:ietf:params:xml:ns:yang:ietf-bfd-mpls-te

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

```
----------------------------------------------------------------
```

This document registers the following YANG modules in the YANG Module Names registry [RFC6020]:

RFC Editor: Replace RFC XXXX with actual RFC number and remove this note.

```
----------------------------------------------------------------
```

Name: iana-bfd-types

Namespace: urn:ietf:params:xml:ns:yang:iana-bfd-types

Prefix: iana-bfd-types

Reference: RFC XXXX

```
----------------------------------------------------------------

----------------------------------------------------------------
```

Name: ietf-bfd-types

Namespace: urn:ietf:params:xml:ns:yang:ietf-bfd-types

Prefix: bfd-types

Reference: RFC XXXX

```
----------------------------------------------------------------

----------------------------------------------------------------
```

Name: ietf-bfd

   Namespace: urn:ietf:params:xml:ns:yang:ietf-bfd

   Prefix: bfd

   Reference: RFC XXXX

   Name: ietf-bfd-ip-sh

   Namespace: urn:ietf:params:xml:ns:yang:ietf-bfd-ip-sh

   Prefix: bfd-ip-sh

   Reference: RFC XXXX

   Name: ietf-bfd-ip-mh

   Namespace: urn:ietf:params:xml:ns:yang:ietf-bfd-ip-mh

   Prefix: bfd-ip-mh

   Reference: RFC XXXX

   Name: ietf-bfd-lag

   Namespace: urn:ietf:params:xml:ns:yang:ietf-bfd-lag

   Prefix: bfd-lag

   Reference: RFC XXXX

   Name: ietf-bfd-mpls

   Namespace: urn:ietf:params:xml:ns:yang:ietf-bfd-mpls

   Prefix: bfd-mpls

   Reference: RFC XXXX

   Name: ietf-bfd-mpls-te

   Namespace: urn:ietf:params:xml:ns:yang:ietf-bfd-mpls-te

   Prefix: bfd-mpls-te

   Reference: RFC XXXX

5.1. IANA-Maintained iana-bfd-types module

   This document defines the initial version of the IANA-maintained
   iana-bfd-types YANG module.

   The iana-bfd-types YANG module mirrors the "BFD Diagnostic Codes"
   registry and "BFD Authentication Types" registry at
   https://www.iana.org/assignments/bfd-parameters/bfd-parameters.xhtml.
   Whenever that registry changes, IANA must update the iana-bfd-types
   YANG module.

6. Acknowledgements

   We would also like to thank Nobo Akiya and Jeff Haas for their
   encouragement on this work. We would also like to thank Rakesh
   Gandhi and Tarek Saad for their help on the MPLS-TE model. We would
   also like to thank Acee Lindem for his guidance.

7. References

7.1. Normative References

   [I-D.ietf-mpls-base-yang]
              Saad, T., Raza, K., Gandhi, R., Liu, X., and V. Beeram, "A
              YANG Data Model for MPLS Base", draft-ietf-mpls-base-
              yang-06 (work in progress), February 2018.

   [I-D.ietf-teas-yang-te]
              Saad, T., Gandhi, R., Liu, X., Beeram, V., Shah, H., and
              I. Bryskin, "A YANG Data Model for Traffic Engineering
              Tunnels and Interfaces", draft-ietf-teas-yang-te-16 (work
              in progress), July 2018.

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

   [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
              DOI 10.17487/RFC3688, January 2004,
              <https://www.rfc-editor.org/info/rfc3688>.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246,
              DOI 10.17487/RFC5246, August 2008,
              <https://www.rfc-editor.org/info/rfc5246>.

   [RFC5586] Bocci, M., Ed., Vigoureux, M., Ed., and S. Bryant, Ed.,
              "MPLS Generic Associated Channel", RFC 5586,
              DOI 10.17487/RFC5586, June 2009,
              <https://www.rfc-editor.org/info/rfc5586>.

   [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection
              (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010,
              <https://www.rfc-editor.org/info/rfc5880>.

   [RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection
              (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881,
              DOI 10.17487/RFC5881, June 2010,
              <https://www.rfc-editor.org/info/rfc5881>.

   [RFC5882] Katz, D. and D. Ward, "Generic Application of
              Bidirectional Forwarding Detection (BFD)", RFC 5882,
              DOI 10.17487/RFC5882, June 2010,
              <https://www.rfc-editor.org/info/rfc5882>.

   [RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection
              (BFD) for Multihop Paths", RFC 5883, DOI 10.17487/RFC5883,
              June 2010, <https://www.rfc-editor.org/info/rfc5883>.

   [RFC5884] Aggarwal, R., Kompella, K., Nadeau, T., and G. Swallow,
              "Bidirectional Forwarding Detection (BFD) for MPLS Label
              Switched Paths (LSPs)", RFC 5884, DOI 10.17487/RFC5884,
              June 2010, <https://www.rfc-editor.org/info/rfc5884>.

   [RFC5885] Nadeau, T., Ed. and C. Pignataro, Ed., "Bidirectional
              Forwarding Detection (BFD) for the Pseudowire Virtual
              Circuit Connectivity Verification (VCCV)", RFC 5885,
              DOI 10.17487/RFC5885, June 2010,
              <https://www.rfc-editor.org/info/rfc5885>.

   [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for
              the Network Configuration Protocol (NETCONF)", RFC 6020,
              DOI 10.17487/RFC6020, October 2010,
              <https://www.rfc-editor.org/info/rfc6020>.

   [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,
              and A. Bierman, Ed., "Network Configuration Protocol
              (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
              <https://www.rfc-editor.org/info/rfc6241>.

   [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure
              Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011,
              <https://www.rfc-editor.org/info/rfc6242>.

   [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration
              Protocol (NETCONF) Access Control Model", RFC 6536,
              DOI 10.17487/RFC6536, March 2012,
              <https://www.rfc-editor.org/info/rfc6536>.

   [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types",
              RFC 6991, DOI 10.17487/RFC6991, July 2013,
              <https://www.rfc-editor.org/info/rfc6991>.

   [RFC7130] Bhatia, M., Ed., Chen, M., Ed., Boutros, S., Ed.,
              Binderberger, M., Ed., and J. Haas, Ed., "Bidirectional
              Forwarding Detection (BFD) on Link Aggregation Group (LAG)
              Interfaces", RFC 7130, DOI 10.17487/RFC7130, February
              2014, <https://www.rfc-editor.org/info/rfc7130>.

   [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
              Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017,
              <https://www.rfc-editor.org/info/rfc8040>.

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

   [RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J.
              Zhang, "YANG Data Model for Key Chains", RFC 8177,
              DOI 10.17487/RFC8177, June 2017,
              <https://www.rfc-editor.org/info/rfc8177>.

   [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams",
              BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018,
              <https://www.rfc-editor.org/info/rfc8340>.

   [RFC8343] Bjorklund, M., "A YANG Data Model for Interface
              Management", RFC 8343, DOI 10.17487/RFC8343, March 2018,
              <https://www.rfc-editor.org/info/rfc8343>.

   [RFC8344] Bjorklund, M., "A YANG Data Model for IP Management",
              RFC 8344, DOI 10.17487/RFC8344, March 2018,
              <https://www.rfc-editor.org/info/rfc8344>.

   [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for
              Routing Management (NMDA Version)", RFC 8349,
              DOI 10.17487/RFC8349, March 2018,
              <https://www.rfc-editor.org/info/rfc8349>.

7.2. Informative References

   [I-D.ietf-lime-yang-connectionless-oam]
              Kumar, D., Wang, Z., Wu, Q., Rahman, R., and S. Raghavan,
              "Generic YANG Data Model for the Management of Operations,
              Administration, and Maintenance (OAM) Protocols that use
              Connectionless Communications", draft-ietf-lime-yang-
              connectionless-oam-18 (work in progress), November 2017.

   [I-D.ietf-rtgwg-lne-model]
              Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X.
              Liu, "YANG Model for Logical Network Elements", draft-
              ietf-rtgwg-lne-model-10 (work in progress), March 2018.

   [I-D.ietf-rtgwg-ni-model]
              Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X.
              Liu, "YANG Model for Network Instances", draft-ietf-rtgwg-
              ni-model-12 (work in progress), March 2018.

   [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K.,
              and R. Wilton, "Network Management Datastore Architecture
              (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018,
              <https://www.rfc-editor.org/info/rfc8342>.

Appendix A. Echo function configuration example

   As mentioned in Section 2.1.2, the mechanism to start and stop the
   echo function, as defined in [RFC5880] and [RFC5881], is
   implementation specific. In this section we provide an example of
   how the echo function can be implemented via configuration.

```
  module: example-bfd-echo
    augment /rt:routing/rt:control-plane-protocols
              /rt:control-plane-protocol/bfd:bfd/bfd-ip-sh:ip-sh
              /bfd-ip-sh:sessions:
      +--rw echo {bfd-types:echo-mode}?
         +--rw desired-min-echo-tx-interval?    uint32
         +--rw required-min-echo-rx-interval?   uint32
```

A.1. Example YANG module for BFD echo function configuration

```
module example-bfd-echo {
  namespace "tag:example.com,2018:example-bfd-echo";

  prefix "example-bfd-echo";

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

  import ietf-bfd {
    prefix "bfd";
  }

  import ietf-bfd-ip-sh {
    prefix "bfd-ip-sh";
  }

  import ietf-routing {
    prefix "rt";
  }

  organization "IETF BFD Working Group";

  contact
    "WG Web:   <http://tools.ietf.org/wg/bfd>
     WG List: <rtg-bfd@ietf.org>

     Editors:  Reshad Rahman (rrahman@cisco.com),
               Lianshu Zheng (vero.zheng@huawei.com),
               Mahesh Jethanandani (mjethanandani@gmail.com)";

  description
    "This module contains an example YANG augmentation for configuration
     of BFD echo function.

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

```
      Redistribution and use in source and binary forms, with or
      without modification, is permitted pursuant to, and subject
      to the license terms contained in, the Simplified BSD License
      set forth in Section 4.c of the IETF Trust's Legal Provisions
      Relating to IETF Documents
      (http://trustee.ietf.org/license-info).

      This version of this YANG module is part of RFC XXXX; see
      the RFC itself for full legal notices.";

  revision 2018-08-01 {
    description "Initial revision.";
    reference
      "RFC XXXX: A YANG data model example augmentation for BFD echo
       function";
  }

  // RFC Ed.: replace XXXX with actual RFC number and remove this
  // note

  /*
   * Groupings
   */
  grouping echo-cfg-parms {
    description "BFD grouping for echo config parameters";
    leaf desired-min-echo-tx-interval {
      type uint32;
      units microseconds;
      default 0;
      description
        "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 BFD [RFC5880] is disabled.";
    }

    leaf required-min-echo-rx-interval {
      type uint32;
      units microseconds;
      default 0;
      description
        "This is the Required Min Echo RX Interval as defined in BFD
         [RFC5880].";
    }
  }

  augment "/rt:routing/rt:control-plane-protocols/"
        + "rt:control-plane-protocol/bfd:bfd/bfd-ip-sh:ip-sh/"
        + "bfd-ip-sh:sessions" {
```

```
      description "Augmentation for BFD echo function.";

      container echo {
        if-feature bfd-types:echo-mode;

        description "BFD echo function container";

        uses echo-cfg-parms;
      }
    }
}
```

Appendix B. Change log

   RFC Editor: Remove this section upon publication as an RFC.

B.1. Changes between versions -16 and -17

   o Addressed IESG comments.

B.2. Changes between versions -15 and -16

   o Added list of modules for YANG module registry.

B.3. Changes between versions -14 and -15

   o Added missing ietf-bfd-types in XML registry.

B.4. Changes between versions -13 and -14

   o Addressed missing/incorrect references in import statements.

B.5. Changes between versions -12 and -13

   o Updated references for drafts which became RFCs recently.

B.6. Changes between versions -11 and -12

   o Addressed comments from YANG Doctor review of rev11.

B.7. Changes between versions -10 and -11

   o Added 2 examples.

   o Added a container around some lists.

   o Fixed some indentation nits.

B.8. Changes between versions -09 and -10

   o Addressed comments from YANG Doctor review.

   o Addressed comments from WGLC.

B.9. Changes between versions -08 and -09

   o Mostly cosmetic changes to abide by draft-ietf-netmod-rfc6087bis.

   o Specified yang-version 1.1.

   o Added data model examples.

   o Some minor changes.

B.10. Changes between versions -07 and -08

   o Timer intervals in client-cfg-parms are not mandatory anymore.

   o Added list of interfaces under "ip-sh" node for authentication
      parameters.

   o Renamed replay-protection to meticulous.

B.11. Changes between versions -06 and -07

   o New ietf-bfd-types module.

   o Grouping for BFD clients to have BFD multiplier and interval
      values.

   o Change in ietf-bfd-mpls-te since MPLS-TE model changed.

   o Removed bfd- prefix from many names.

B.12. Changes between versions -05 and -06

   o Adhere to NMDA-guidelines.

   o Echo function config moved to appendix as example.

   o Added IANA YANG modules.

   o Addressed various comments.

B.13. Changes between versions -04 and -05

   o "bfd" node in augment of control-plane-protocol.

   o Removed augment of network-instance. Replaced by schema-mount.

   o Added information on interaction with other YANG modules.

B.14. Changes between versions -03 and -04

   o Updated author information.

   o Fixed YANG compile error in ietf-bfd-lag.yang which was due to
      incorrect when statement.

B.15. Changes between versions -02 and -03

   o Fixed YANG compilation warning due to incorrect revision date in
      ietf-bfd-ip-sh module.

B.16. Changes between versions -01 and -02

   o Replace routing-instance with network-instance from YANG Network
      Instances [I-D.ietf-rtgwg-ni-model]

B.17. Changes between versions -00 and -01

   o Remove BFD configuration parameters from BFD clients, all BFD
      configuration parameters in BFD

   o YANG module split in multiple YANG modules (one per type of
      forwarding path)

   o For BFD over MPLS-TE we augment MPLS-TE model

   o For BFD authentication we now use YANG Data Model for Key Chains
      [RFC8177]

Authors' Addresses

   Reshad Rahman (editor)
   Cisco Systems
   Canada

   Email: rrahman@cisco.com

   Lianshu Zheng (editor)
   Huawei Technologies
   China

   Email: vero.zheng@huawei.com

   Mahesh Jethanandani (editor)
   Xoriant Corporation
   1248 Reamwood Ave
   Sunnyvale, California 94089
   USA

   Email: mjethanandani@gmail.com

   Santosh Pallagatti
   Rtbrick
   India

   Email: santosh.pallagatti@gmail.com

   Greg Mirsky
   ZTE Corporation

   Email: gregimirsky@gmail.com

Network Working Group   M. Jethanandani
Internet-Draft   S. Agarwal
Intended status: Standards Track   Cisco Systems, Inc
Expires: August 26, 2017   A. Mishra
                                                      A. Saxena
                                                 Ciena Corporation
                                                        A. Dekok
                                            Network RADIUS SARL
                                                February 22, 2017

                       Secure BFD Sequence Numbers
               draft-sonal-bfd-secure-sequence-numbers-00

Abstract

   This document describes a security enhancements for the BFD packet's
   sequence number.

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

Copyright Notice

Table of Contents

1. Introduction

   BFD [RFC5880] section 6.7 describes the use of monotonically
   incrementing 32-bit sequence numbers for use in authentication of BFD
   packets. While this method protects against simple replay attacks,
   the monotonically incrementing sequence numbers are predictable and
   vulnerable to more complex attack vectors. This document proposes
   the use of non-monotonically-incrementing sequence numbers in BFD
   authentication TLVs to enhance the security of BFD sessions.
   Specifically, the document presents a method to generate pseudo-
   random sequence numbers on the frame by algorithmically hashing
   monotonically increasing sequence numbers. Further security may be
   introduced by resetting un-encrypted sequence to a random value when
   the 32-bit sequence number rolls-over.

2. Theory of operations

   Instead of monotonically increasing the sequence number or even
   occasionally monotonically increasing the sequence number, the next
   sequence number is generated by computing a hash on what would have
   been the next sequence number using a shared key. That computed hash
   is then inserted into the sequence number field of the packet. In
   case of BFD Authentication [I-D.ietf-bfd-optimizing-authentication],
   the sequence number used in computing an authenticated packet would
   be this new computed hash. Even though the BFD Authentication

[I-D.ietf-bfd-optimizing-authentication] sequence number is
independent of this enhancement, it would benefit by using the
computed hash.

A normal BFD packet with authentication will undergo the following
steps, where:

[O]: original RFC 5880 packet with monotonically increasing sequence
number

[S]: psuedo random sequence number

[A]: Authentication

```
            Sender                    Receiver

            [O] [S] [A] ------------- [A] [S] [O]
```

In order to encode a sequence number, the sender would identify a
hash algorithm (symmetric) that would create a 32 bit hash. The
hashing key is provisioned securely on the sender and receiver of the
BFD session. The mechanism of provisioning such a key is outside the
scope of this draft. Instead of using the sequence number, the
sender encodes the sequence number with the hashing key to produce a
hash. Upon receiving the BFD Control packet, the receiver decodes
the hash with the provisioned hashing key by performing a reverse
hash. Note: The first sequence number can be obtained using the same
logic as the My Discriminator value.

k: hashing key

s: sequence number

O: original RFC 5880 packet with monotonically increasing sequence
number

R: remainder of packet

H1: hash of s

H2: hash of entire packet

A: H2 + insertion in packet

hash(s, k) = H1

hash((H1 + R), k) = H2

   hash'((Packet - H2), k) == H2 ? Good packet : bad packet

   hash'(H1, k) == s ? Good sequence number : bad sequence number

                    Sender   Receiver

                    [O] [H1] [A] -------- [A] [H1] [O]

3. Impact of using a hash

   Under this proposal, every packet's sequence number is encoded within
   a hash. Therefore there is some impact on the system and its
   performance while encoding/decoding the hash. As security measures
   go, this enhancement greatly increases the security of the packet
   with or without authentication of the entire packet.

4. IANA Considerations

   This document makes no request of IANA.

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

5. Security Considerations

6. Acknowledgements

7. References

7.1. Normative References

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

   [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection
              (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010,
              <http://www.rfc-editor.org/info/rfc5880>.

7.2. Informative References

   [I-D.ietf-bfd-optimizing-authentication]
              Jethanandani, M., Mishra, A., Saxena, A., and M. Bhatia,
              "Optimizing BFD Authentication", draft-ietf-bfd-
              optimizing-authentication-02 (work in progress), January
              2017.

Authors' Addresses

   Mahesh Jethanandani
   Cisco Systems, Inc
   170 West Tasman Drive
   San Jose, CA 95070
   USA

   Email: mjethanandani@gmail.com

   Sonal Agarwal
   Cisco Systems, Inc
   170 W. Tasman Drive
   San Jose, CA 95070
   USA

   Email: agarwaso@cisco.com
   URI: www.cisco.com

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

   Email: mishra.ashesh@gmail.com

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

   Email: ankurpsaxena@gmail.com

   Alan DeKok
   Network RADIUS SARL
   100 Centrepointe Drive #200
   Ottowa, ON K2G 6B1
   Canada

   Email: aland@networkradious.com

          Bidirectional Forwarding Detection (BFD) on Multi-chassis Ling
              Aggregation Group (MC-LAG) Interfaces in IP Networks
                    draft-tanmir-rtgwg-bfd-mc-lag-ip-01

   Abstract

      This document describes use of Bidirectional Forwarding Detection for
      Multi-chassis Link Aggregation Group to provide faster than Link
      Aggregation Control Protocol convergence. This specification
      enhances and updates RFC 7130 "Bidirectional Forwarding Detection
      (BFD) on Link Aggregation Group (LAG) Interfaces".

Table of Contents

1. Introduction

   The [RFC7130] defines use of Bidirectional Forwarding Detection (BFD)
   on Link Aggregation Group (LAG) interfaces. Multi-chassis LAG (MC-
   LAG) is type of LAG [IEEE.802.1AX.2008] with member links terminated
   on separate chassis. [IEEE.802.1AX.2008] does not specify MC-LAG but
   doesn't preclude it either. Link Aggregation Control Protocol
   (LACP), also defined in [IEEE.802.1AX.2008], can work with MC-LAG
   but, as in LAG case, can detect link failure only in range of single
   seconds. This document defines how mechanism defined to work on LAG
   interfaces [RFC7130] can be adapted to MC-LAG case to enable sub-
   second detection of member link failure.

1.1. Conventions used in this document

1.1.1. Terminology

   BFD: Bidirectional Forwarding Detection

   LAG: Link Aggregation Group

   LACP: Link Aggregation Control Protocol

   MC-LAG: Multi-chassis Link Aggregation Group

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

   [RFC7130] does not specify selection of the destination IP address
   for the BFD control packet. The only requirement related to the
   selection is in Section 2.1 stating that the use of address family
   across all member links of the given LAG MUST be consistent across
   all the links. Thus it is implied that the same unicast IP address
   will be used on all member links of the LAG as use of different
   destination addresses would defeat the purpose of [RFC7130]
   transforming the case into set of single-hop BFD sessions [RFC5881].
   But single unicast IP address may not work in MC-LAG case as the
   member links are terminated on the separate chassis. This document
   proposes how to overcome this problem if using IP or Multi-Protocol
   Label Switching (MPLS) data plane encapsulation.

3. BFD on MC-LAG with IP only data plane

   As described in [RFC7130] micro-BFD session on the LAG interfaces may
   use either IPv4 or IPv6 address family. In some cases two sessions,
   one with IPv4 and one with IPv6 addresses, may run concurrently.
   This document doesn't change any of these but specifies selection of
   the destination IP address in MC-LAG use case:

   o if IPv4 address family being used for micro-BFD session, then the
      link-local multicast address 224.0.0.0/24 SHOULD be used as the
      destination IP address. Subnet broadcast address MAY be used as
      the destination IP address as well;

   o if the address family used is IPv6, then the IPv6 link-local
      multicast address FF02:0:0:0:0:0:0:2 MUST be used as the
      destination IP address.

4. IANA Considerations

   This document makes no requests for IANA allocations. This section
   may be deleted by RFC Editor.

5. Security Considerations

   Security considerations discussed in [RFC7130] apply to this
   document.

6. Acknowledgements

7. Normative References

   [IEEE.802.1AX.2008]
              "IEEE Standard for Local and metropolitan area networks -
              Link Aggregation", IEEE 802.1-AX, November 2008.

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

   [RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection
              (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881,
              DOI 10.17487/RFC5881, June 2010,
              <http://www.rfc-editor.org/info/rfc5881>.

   [RFC7130] Bhatia, M., Ed., Chen, M., Ed., Boutros, S., Ed.,
              Binderberger, M., Ed., and J. Haas, Ed., "Bidirectional
              Forwarding Detection (BFD) on Link Aggregation Group (LAG)
              Interfaces", RFC 7130, DOI 10.17487/RFC7130, February
              2014, <http://www.rfc-editor.org/info/rfc7130>.

Authors' Addresses

   Greg Mirsky
   ZTE Corp.

   Email: gregimirsky@gmail.com

   Jeff Tantsura
   Indvidual

   Email: jefftant.ietf@gmail.com

       Bidirectional Forwarding Detection (BFD) on Multi-chassis Ling
         Aggregation Group (MC-LAG) Interfaces in IP/MPLS Networks
                  draft-tanmir-rtgwg-bfd-mc-lag-mpls-01

Abstract

   This document describes use of Bidirectional Forwarding Detection for
   Multi-chassis Link Aggregation Group to provide faster than Link
   Aggregation Control Protocol convergence. This specification
   enhances and updates RFC 7130 "Bidirectional Forwarding Detection
   (BFD) on Link Aggregation Group (LAG) Interfaces".

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 11, 2017.

the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction

   The [RFC7130] defines use of Bidirectional Forwarding Detection (BFD)
   on Link Aggregation Group (LAG) interfaces. Multi-chassis LAG (MC-
   LAG) is type of LAG [IEEE.802.1AX.2008] with member links terminated
   on separate chassis. [IEEE.802.1AX.2008] does not specify MC-LAG but
   doesn't preclude it either. Link Aggregation Control Protocol
   (LACP), also defined in [IEEE.802.1AX.2008], can work with MC-LAG
   but, as in LAG case, can detect link failure only in range of single
   seconds. This document defines how mechanism defined to work on LAG
   interfaces [RFC7130] can be adapted to MC-LAG case to enable sub-
   second detection of member link failure.

1.1. Conventions used in this document

1.1.1. Terminology

   ACH: Associated Channel Header

   BFD: Bidirectional Forwarding Detection

   BoS: Bottom of the Stack

   G-ACh: Generic Associated Channel

   GAL: Generic Associated Label

   LAG: Link Aggregation Group

   LACP: Link Aggregation Control Protocol

    MC-LAG: Multi-chassis Link Aggregation Group

    MPLS: Multi-Protocol Label Switching

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

    [RFC7130] does not specify selection of the destination IP address
    for the BFD control packet. The only requirement related to the
    selection is in Section 2.1 stating that the use of address family
    across all member links of the given LAG MUST be consistent across
    all the links. Thus it is implied that the same unicast IP address
    will be used on all member links of the LAG as use of different
    destination addresses would defeat the purpose of [RFC7130]
    transforming the case into set of single-hop BFD sessions [RFC5881].
    But single unicast IP address may not work in MC-LAG case as the
    member links are terminated on the separate chassis. This document
    proposes how to overcome this problem if using IP or Multi-Protocol
    Label Switching (MPLS) data plane encapsulation.

3. BFD on MC-LAG with IP/MPLS data plane

    There are more optional encapsulation formats for the case of micro-
    BFD on MC-LAG over IP/MPLS data plane:

    o [RFC5586] defined special purpose Generic Associated channel Label
       (GAL) that MAY be used in MPLS encapsulation of the micro-BFD
       control packet over MPSL data plane. Depending on the channel
       type specified in the Associated Channel Header (ACH) that
       immediately follows after the GAL, micro-BFD MAY use IP/UDP, as
       displayed in Figure 1 or BFD format, i.e. BFD control packet
       without IP and UDP headers.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      GAL                  | TC |1|    TTL     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 0 1|0 0 0 0|   Reserved  |     IPv4 channel (0x0021)      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination IP address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Source IP address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      UDP header                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                   BFD Control Packet                          ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
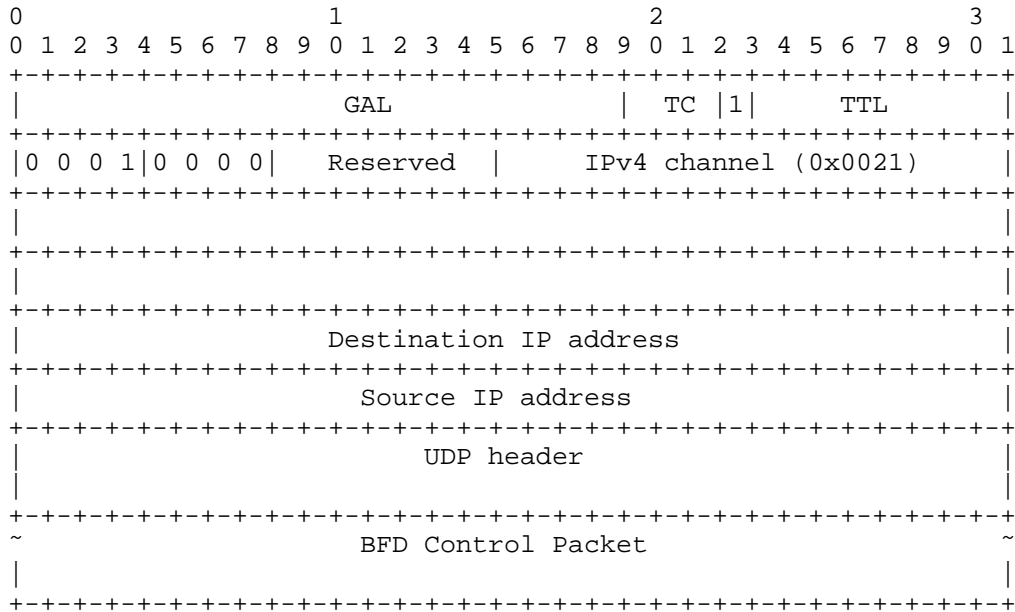
Figure 1: BFD on MC-LAG member link on IPv4/MPLS data plane

If IP/UDP format of BFD over MC-LAG interfaces is used, then for IPv4
address family the destination IP address MUST be selected from 127/8
range [RFC4379], and if IPv6 address family is used, then the
destination IP address MUST be selected from 0:0:0:0:0:FFFF:127/104
range.

4. IANA Considerations

This document makes no requests for IANA allocations. This section
may be deleted by RFC Editor.

5. Security Considerations

Security considerations discussed in [RFC7130] apply to this
document.

6. Acknowledgements

7. Normative References

[IEEE.802.1AX.2008]
           "IEEE Standard for Local and metropolitan area networks -
           Link Aggregation", IEEE 802.1-AX, November 2008.

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

   [RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol
              Label Switched (MPLS) Data Plane Failures", RFC 4379,
              DOI 10.17487/RFC4379, February 2006,
              <http://www.rfc-editor.org/info/rfc4379>.

   [RFC5586] Bocci, M., Ed., Vigoureux, M., Ed., and S. Bryant, Ed.,
              "MPLS Generic Associated Channel", RFC 5586,
              DOI 10.17487/RFC5586, June 2009,
              <http://www.rfc-editor.org/info/rfc5586>.

   [RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection
              (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881,
              DOI 10.17487/RFC5881, June 2010,
              <http://www.rfc-editor.org/info/rfc5881>.

   [RFC7130] Bhatia, M., Ed., Chen, M., Ed., Boutros, S., Ed.,
              Binderberger, M., Ed., and J. Haas, Ed., "Bidirectional
              Forwarding Detection (BFD) on Link Aggregation Group (LAG)
              Interfaces", RFC 7130, DOI 10.17487/RFC7130, February
              2014, <http://www.rfc-editor.org/info/rfc7130>.

Authors' Addresses

   Greg Mirsky
   ZTE Corp.

   Email: gregimirsky@gmail.com

   Jeff Tantsura
   Individual

   Email: jefftant.ietf@gmail.com