Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 2, 2019

R. Rahman, Ed.
Cisco Systems
L. Zheng, Ed.
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

- Addressed IESG comments.

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

- Added list of modules for YANG module registry.

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

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

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

- Addressed missing/incorrect references in import statements.

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

- Updated references for drafts which became RFCs recently.

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

- Addressed comments from YANG Doctor review of rev11.

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

- Added 2 examples.

- Added a container around some lists.

- Fixed some indentation nits.

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

- Addressed comments from YANG Doctor review.

- Addressed comments from WGLC.

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

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

- Specified yang-version 1.1.

- Added data model examples.

- Some minor changes.

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

- Timer intervals in client-cfg-parms are not mandatory anymore.

- Added list of interfaces under "ip-sh" node for authentication parameters.

- Renamed replay-protection to meticulous.

### B.11. Changes between versions -06 and -07

- New ietf-bfd-types module.

- Grouping for BFD clients to have BFD multiplier and interval values.

- Change in ietf-bfd-mpls-te since MPLS-TE model changed.

- Removed bfd- prefix from many names.

### B.12. Changes between versions -05 and -06

- Adhere to NMDA-guidelines.

- Echo function config moved to appendix as example.

- Added IANA YANG modules.

- Addressed various comments.

### B.13. Changes between versions -04 and -05

- "bfd" node in augment of control-plane-protocol.
- Removed augment of network-instance. Replaced by schema-mount.
- Added information on interaction with other YANG modules.

### B.14. Changes between versions -03 and -04

- Updated author information.
- Fixed YANG compile error in ietf-bfd-lag.yang which was due to incorrect when statement.

### B.15. Changes between versions -02 and -03

- Fixed YANG compilation warning due to incorrect revision date in ietf-bfd-ip-sh module.

### B.16. Changes between versions -01 and -02

- Replace routing-instance with network-instance from YANG Network Instances [I-D.ietf-rtgwg-ni-model]

### B.17. Changes between versions -00 and -01

- Remove BFD configuration parameters from BFD clients, all BFD configuration parameters in BFD
- YANG module split in multiple YANG modules (one per type of forwarding path)
- For BFD over MPLS-TE we augment MPLS-TE model
- 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

NVO3 WG
Internet-Draft
Intended status: Standards Track
Expires: January 2, 2017

E. Nordmark
C. Appanna
A. Lo
Arista Networks
S. Boutros
A. Dubey
VMware

Jul 2016

# Layer-Transcending Traceroute for Overlay Networks like VXLAN
draft-nordmark-nvo3-transcending-traceroute-03

## Abstract

Tools like traceroute have been very valuable for the operation of the Internet. Part of that value comes from being able to display information about routers and paths over which the user of the tool has no control, but the traceroute output can be passed along to someone else that can further investigate or fix the problem.

In overlay networks such as VXLAN and NVGRE the prevailing view is that since the overlay network has no control of the underlay there needs to be special tools and agreements to enable extracting traces from the underlay. We argue that enabling visibility into the underlay and using existing tools like traceroute has been overlooked and would add value in many deployments of overlay networks.

This document specifies an approach that can be used to make traceroute transcend layers of encapsulation including details for how to apply this to VXLAN. The technique can be applied to other encapsulations used for overlay networks. It can also be implemented using current commercial silicon.

## Status of this Memo

This Internet-Draft will expire on January 2, 2017.

Copyright Notice

Table of Contents

## 1. Introduction

Tools like traceroute have been very valuable for the operation of the Internet. Part of that value comes from being able to display information about routers and paths over which the user of the tool has no control, but the traceroute output can be passed along to someone else that can further investigate or fix the problem. The output of traceroute can be included in an email or a trouble ticket to report the problem. This provide a lot more information than the mere indication that A can’t communicate with B, in particular when the failures are transient. The ping tool provides some of the same benefits in being able to return ICMP errors such as host unreachable messages.

This document shows how those tools can be used to gather information for both the overlay and underlay parts of an end-to-end path by providing the option to have some packets use a uniform time-to-live (ttl) model for the tunnels, and associated ICMP error handling. These changes are limited to the tunnel ingress and egress points.

The desire to make traceroute provide useful information for overlay network is not an argument against also using a layered approach for OAM as specified in e.g., [I-D.tissa-lime-yang-oam-model]. Such approaches are quite appropriate for continuos monitoring at different layers and across different domains. A layer transcending traceroute complements the ability to do layered and/or continuos monitoring.

The traceroute tool relies on receiving ICMP errors [RFC0792] in combination with using different IP time-to-live values. That results in the packet making it further and further towards the destination with ICMP ttl exceeded errors being received from each hop. That provides the user the working path even if the packets are black holed eventually, and also provides any errors like ICMP host unreachable. The fundamental assumption is that the ttl is decremented for each hop and that the resulting ICMP ttl exceeded errors are delivered back to the host.

When some encapsulation is used to tunnel packets there is an architectural question how those tunnels should be viewed from the rest of the network. Different models were described first for diffserv in [RFC2983] and then applied to MPLS in [RFC3270] and expanded to MPLS ttl handling in [RFC3443] and those models apply to other forms of direct or indirect IP in IP tunnels. Those RFCs define two models for ttl that are of interest to us:

- o A pipe model, where the tunnel is invisible to the rest of the network in that it looks like a direct connection between the tunnel ingress and egress.

- A uniform model, where the ttl decrements uniformly for hops outside and inside the tunnel.

The tunneling mechanisms discussed in NVO3 (such as VXLAN [RFC7348], NVGRE [I-D.sridharan-virtualization-nvgre], GENEVE [I-D.gross-geneve], and GUE [I-D.herbert-gue]), have either been specified to provide the pipe model of a tunnel or are silent on the setting of the outer ttl. Those protocols can be extended to have an optional uniform tunnel model when the payload is IP, following the same model as in [RFC3443]. Note that these encapsulations carry Ethernet frames hence are not even aware that the payload is IP. However, IP is the bulk of what is carried over such tunnels and the ingress NVE can inspect the IP part of the Ethernet frame.

However, for general application traffic the pipe model is fine and might even be expected by some applications. In general, when the source and destination IP are in the same IP subnet the ttl should not be decremented. Thus it makes sense to have a way to selectively enable the uniform model perhaps based on some method to identify packets associated with traceroute or some marker in the packet itself that the traceroute tool can set.

## 2. Solution Overview

The pieces needed to accomplish this are:

- One or more ways to select the uniform model packets at the tunnel ingress.

- Tunnel ingress copying out the original ttl from a selected packet to the outer IP header, and then doing a check and decrement of that ttl.

- If that ttl check results in ttl expiry at the tunnel ingress, then deliver an ICMP ttl exceeded packet back to the host.

- A mechanism by which the tunnel egress knows which packets should have uniform model, for instance a bit in the encapsulation header.

- The tunnel egress copying in the ttl (for identified packets) from the outer header to the inner IP header, then doing a check and decrement of that ttl.

- o If ttl check results in ttl expiry at the tunnel egress, then deliver an ICMP error back to the original host (or, perhaps better, to tunnel ingress the same way as underlay routers do).

- o IP routers in the underlay will deliver any ICMP errors to the source IP address of the packet. For tunneled packets that will be the tunnel ingress. Hence the tunnel ingress needs to be able to take such ICMP errors and form corresponding ICMP errors that are sent back to the host. The requirement in [RFC1812] ensures that the ICMP errors will contain enough headers to form such an ICMP error. It has been noted that there are routers in the Internet which decades later fail to conform to that aspect of [RFC1812].

The idea to reflect (some) ICMP errors from inside a tunnel back to the original source goes back to IPv6 in IPv4 encapsulation as specified in [RFC1933] and [RFC2473]. However, those drafts did not advocate using a uniform ttl model for the tunnels but did handle ICMP packet too big and other unreachable messages. Those drafts specify how to reflect ICMP errors received from underlay routers to ICMP errors sent to the original host. The addition of handling ICMP ttl exceeded errors for uniform tunnel model is straight forward.

The information carried in the ICMP errors are quite limited - the original packet plus an ICMP type and code. However, there are extension mechanisms specified in [RFC4884] and used for MPLS in [RFC4950] which include TLVs with additional information. If there are additional information to include for overlay networks that information could be added by defining new ICMP Extensions Objects based on [RFC4884]. An example of such an extension for ECMP information is included in this document.

## 3. Goals and Requirements

The following goals and requirements apply:

- o No changes needed in the underlay.

- o Optional changes on the decapsulating end.

- o ECMP friendly. If the underlay employs equal cost multipath routing then one should be able to use this mechanism to trace the same path as a given TCP or UDP flow is using. In addition, one should be able to explore different ECMP paths by varying the IP addresses and port numbers in the packets originated by traceroute on the host.

o Provide output which makes it possible to compare a regular overlay traceroute with the layer-transcending output.

## 4. Definition Of Terms

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The terminology such as NVE, and TS are used as specified in [RFC7365]:

o Network Virtualization Edge (NVE): An NVE is the network entity that sits at the edge of an underlay network and implements L2 and/or L3 network virtualization functions.

o Tenant System (TS): A physical or virtual system that can play the role of a host or a forwarding element such as a router, switch, firewall, etc.

o Virtual Access Points (VAPs): A logical connection point on the NVE for connecting a Tenant System to a virtual network.

o Virtual Network (VN): A VN is a logical abstraction of a physical network that provides L2 or L3 network services to a set of Tenant Systems.

o Virtual Network Context (VN Context) Identifier: Field in an overlay encapsulation header that identifies the specific VN the packet belongs to.

We use the VTEP term in [RFC7348] as synonymous with NVE, and VNI as synonymous to VN Context Identifier.

## 5. Example Topologies

The following example topologies illustrate different cases where we want a tracing capability. The examples are for overlay technologies such as VXLAN which provide a layer 2 overlay on IP. The cases for layer 3 overlay on top of IP are simpler and not shown in this document.

The VXLAN term VTEP is used as synonymous to NVO3's NVE term.

```
   -----------               -----------
   |    H1   |               |    H2   |
   | 1.0.1.1 |               | 1.0.1.2 |
   |         |               |         |
   -----------               -----------
        |                         |
        |                         |
   -----------   -----------  -----------
   |  VtepA  |   |    R1   |  |  VtepB  |
   | 2.0.1.1 | --| 2.0.1.2 |  | 2.0.2.1 |
   |         |   | 2.0.2.2 |--|         |
   -----------   -----------  -----------
```

Simple L2 overlay

The figure above shows two hosts connected using an underlay which provides a layer two service. Thus H1 and H2 are in the same subnet and unaware of the existence of the underlay. Thus a normal ping or traceroute would not be able to provide any information about the nature of a failure; either packets get through or they do not. When the packets get through traceroute would output something like:

```
traceroute to 1.0.1.2 (1.0.1.2), 30 hops max, 60 byte packets
 1  1.0.2.1 (1.0.2.1)  1.104 ms  1.235 ms  1.729 ms
```

In this case it would be desirable to be able to traceroute from H1 to H2 (and vice versa) and observe VtepA, R1, VtepB and H2. Thus in the case of packets getting through traceroute would output:

```
traceroute to 1.0.1.2 (1.0.1.2), 30 hops max, 60 byte packets
 1  2.0.1.1 (2.0.1.1)  1.104 ms  1.235 ms  1.729 ms
 2  2.0.1.2 (2.0.1.2)  2.106 ms  2.007 ms  2.156 ms
 3  2.0.2.1 (2.0.2.1)  35.034 ms  24.490 ms  21.626 ms
 4  1.0.1.2 (1.0.1.2)  40.830 ms  44.694 ms  75.620 ms
```

Note that the underlay and overlay might exist in completely separate addressing domains. Thus H1 might not be able to reach any of the underlay addresses. And the underlay IP addresses might overlap the overlay IP addresses. For example, it would be completely valid to see e.g. VtepA having the same IP address as H1. The user of this tool need to understand that the utility of the traceroute output is to get information to determine whether the issue is in the underlay or overlay, and be able to pass the underlay information to the operator of the underlay.

In overlay networks without any ARP/ND optimizations ARP/ND packets would be flooded between the tunnel endpoints. Thus if there is some communication failure between H1 and H2, then H1 above might not have an ARP entry for H2.  This results in traceroute not being able to output any data.  This implies that in order to use traceroute to trouble shoot the issue one would need some workaround, such as installing some temporary ARP entries on the hosts.

```
-----------                  -----------  -----------  -----------
|    H1   |                  |    R2   |  |    R3   |  |    H4   |
| 1.0.1.1 |                  | 1.0.2.2 |--| 1.0.2.3 |  |         |
|         |                  | 1.0.1.2 |  | 1.0.3.3 |--| 1.0.3.4 |
-----------                  -----------  -----------  -----------
     |                            |
     |                            |
-----------   -----------  -----------
| VtepA  |   |    R1   |  |  VtepB  |
| 2.0.1.1 | --| 2.0.1.2 |  | 2.0.2.1 |
|         |   | 2.0.2.2 |--|         |
-----------   -----------  -----------
```

L2 overlay as part of larger network

The figure above has a overlay router the nexthop as seen by H1.  In this case a normal overlay traceroute would be able to display the overlay path i.e.

```
traceroute to H4, 30 hops max, 60 byte packets
 1  R2
 2  R3
 3  H4
```

The layer-transcending traceroute would show the combination of the underlay and overlay paths i.e.,

```
traceroute to H4, 30 hops max, 60 byte packets
 1  VtepA
 2  R1
 3  VtepB
 4  R2
 5  R3
 6  H4
```

```
 -----------           -------------------             -----------
 |    H1   |           |        R5       |             |    H6   |
 | 1.0.1.1 |           |                 |             |         |
 |         |           | 1.0.1.2 1.0.5.5 |             | 1.0.5.6 |
 -----------           |-----------------|             -----------
      |                |    |       |    |                  |
      |                |    |       |    |                  |
 ----------- ----------- |-----------------| ----------- -----------
 | VtepA   | |   R1    | |  VtepB    VtepC | |   R6    | |  VtepD  |
 | 2.0.1.1 |-| 2.0.1.2 | | 2.0.2.1 3.0.1.1 |-| 3.0.1.2 | |         |
 |         | | 2.0.2.2 |-|                 | | 3.0.2.2 |-| 3.0.3.1 |
 ----------- ----------- ------------------- ----------- -----------
```

Multiple L2 overlays in path

The figure above has multiple overlay network segments, that are connected in one router which provides the tunnel endpoints for both overlay segments plus routing for the overlay. A more general picture would be to have an overlay routed path between the two NVEs e.g., VtepB and VtepC connected to different routers in the overlay. However, such a drawing in ASCII art doesn’t fit on the page.

An normal overlay traceroute in the above topology would show the overlay router i.e.,

```
traceroute to H6, 30 hops max, 60 byte packets
 1  R5
 2  H6
```

The layer-transcending traceroute would show the combination of the underlay and overlay paths i.e.,

```
traceroute to H6, 30 hops max, 60 byte packets
 1  VtepA
 2  R1
 3  VtepB
 4  R5
 5  VtepC
 6  R6
 7  VtepD
 8  H6
```

Note that the R3 device, which include VtepB and VtepC, appears as three hops in the traceroute output. That is needed to be able to correlate the output with the overlay output which has R3. That correlation would be hard if the R3 device only appeared as VtepB in the LTTON output. The three-hop representation also stays invariant whether or not the NVEs and overlay router are implemented by a single device or multiple devices.

## 6. Controlling and selecting ttl behavior

The network admin needs to be able to control who can use the layer transcending traceroute, since the operator might not want to disclose the underlay topology to all its users all the time. There are different approaches for this such as designating particular ports (Virtual Access Points in NVO3 terminology) on a NVE to have uniform ttl tunnel model. We have found it useful to be able to enable this capability on a per port and/or virtual network basis, in addition to having a global setting per NVE.

When enabled on the NVEs the user on the TS needs to be able to control which traffic is subject to which tunnel mode. The normal traffic would use the pipe ttl tunnel model and only explicit trace applications are likely to want to use the uniform ttl tunnel model. Hence it makes sense to use some marker in the packets sent by the TS to select those packets for uniform model on the NVE. Such a mechanism should usable so that the user can perform both a regular traceroute and a LTTON.

Potentially different fields in the packets originated by traceroute on the TS can be used to mark the packets for uniform ttl tunnel model. However, many of those fields such as source and destination port numbers and protocol might be used in hashing for ECMP. The marking that can be used without impacting ECMP is the DSCP field in the packet. That field can be set with an option (--tos) in at least some existing traceroute implementations.

Note that when DSCP is used for such marking it is a configured choice subject to agreement between the operator of the TS and NVE. The matching on the NVE should ignore the ECN bits as to not interfere with ECN.

However, the DSCP value used in the overlay might have an impact on the forwarding of the packets. In such a case one can use an alternative selector such as the UDP source port number. That has the downside of affecting the has values used for ECMP and link aggregation port selection.

## 7. Introducing a ttl copyin flag in the encapsulation header

When this approach is applied to VXLAN [RFC7348] the decapsulating NVE has to be able to identify packets that have to be processed in the uniform ttl tunnel model way. For that purpose we define a new flag which is sent by the encapsulating NVE on selected packets, and is used by the decapsulating NVE to perform the ttl copyin, decrement and check.

In addition to the one I-flag defined in [RFC7348] we define a new T-flag to capture this the trace behavior at the decapsulating tunnel endpoint.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|R|R|R|I|R|R|T|            Reserved                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                VXLAN Network Identifier (VNI) |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

New fields:

T-flag: When set indicates that decapsulator should take the outer ttl and copy it to the inner ttl, and then check and decrement the resulting ttl.

## 8. Encapsulation Behavior

If the uniform ttl model is enabled for the input, and the received naked packet matches the selector, then the ingress NVE will perform these additional operations as part of encapsulating an IPv4 or IPv6 packet:

o Examine the IPv4 TTL (or IPv6 hopcount, respectively) on receipt and if 1 or less, then drop the packet and send an ICMPv4 (or ICMPv6) ttl exceeded back to the original host. Since the NVE is operating on a L2 packet, it might not have any layer 3 interfaces or routes for the originating host. Thus it sends the packet back to the source L2 address of the packet back out the ingress port - without any IP address lookup.

o If ttl did not expire, then decrement the above ttl/hopcount and place it in the outer IP header. Encapsulate and send the packet as normal.

o If some other errors prevent sending the packet (such as unknown VN Context Id, no flood list configured), then the NVE SHOULD send an ICMP host unreachable back to the host.

The ingress NVE will receive ICMP errors from underlay routers and the egress NVE; whether due to ttl exceeded or underlay issues such as host unreachable, or packet too big errors. The NVE should take such errors, and in addition to any local syslog etc, generate an ICMP error sent back to the host. The principle for this is specified in [RFC1933] and [RFC2473]. Just like in those specifications, for the inner and outer IP header could be off different version. A common case of that might be an IPv6 overlay with an IPv4 underlay. That case requires some changes in the ICMP type and code values in addition to recreating the packets. The place where LTTON differs from those specifications is that there is an NVO3 header and (for L2 over L3) and L2 header in the packet.

The figures below show an example of ICMP header re-generation at VtepA for the case of IPv6 overlay with IPv4 underlay. The case of IPv4 over IPv4 is similar and simpler since the ICMP header is the same for both overlay and underlay. The example uses VXLAN encapsulation to provide the concrete details, but the approach applies to other NVO3 proposals.

```
                +--------------+
                | IPv4 Header  |
                | src = R1     |
                | dst = VtepA  |
                +--------------+
                |    ICMPv4    |
                |    Header    |
                |   type = X   |
                |   code = Y   |
         - -    +--------------+
                | IPv4 Header  |
                | src = VtepA  |
        IPv4    | dst = VtepB  |
                +--------------+
       Packet   |      UDP     |
                | dst = VXLAN  |
         in     +--------------+
                |   Ethernet   |
       Error    | DA = H2 mac  |
                | SA = H1 mac  |
                +--------------+   - -
                |     IPv6     |
                | src = H1 ipv6|
                | dst = H2 ipv6|   Original IPv6
                +--------------+   Packet.
                |  Transport   |   Used to
                |    Header    |   generate an
                +--------------+   ICMPv6
                |              |   error message
                ~     Data     ~   back to the source.
                |              |
         - -    +--------------+   - -
```

ICMPv4 Error Message Returned to Encapsulating Node

The above underlay ICMPv4 is used to form an overlay ICMPv6 packet by extracting the Ethernet DA from the inner Ethernet SA, and forming an IPv6 header where the source address is based on the source address of the ICMPv4 error. The ICMPv6 type and code values are set based on the ICMPv4 type and code values.

```
                 +--------------+
                 |   Ethernet   |
                 | DA = H1 mac  |   From ICMPv4 packet
                 | SA = VtepA   |   in error
                 +--------------+
                 | IPv6 Header  |
                 | src = ::R1   |   96 zeros + IPv4 address
                 | dst = H1 ipv6|
                 +--------------+
                 |    ICMPv6    |
                 |    Header    |
                 |   type = X’  |   Type and code mapped
                 |   code = Y’  |   from v4 to v6 values
          - -    +--------------+   - -
                 |     IPv6     |
         IPv6    | src = H1 ipv6|
                 | dst = H2 ipv6|   Unmodified from
        Packet   +--------------+   ICMPv4 error
                 |  Transport   |
          in     |    Header    |
                 +--------------+
        Error    |              |
                 ~     Data     ~
                 |              |
          - -    +--------------+   - -
```

Generated ICMPv6 Error Message for Overlay Source

In the case of IPv6 over IPv4 the above example setting of the IPv6 source address results in this type of traceroute output:

```
traceroute to 2000:0:0:40::2, 30 hops max, 80 byte packets
 1  ::2.0.1.1 (::2.0.1.1)  1.231 ms  1.004 ms  1.126 ms
 2  ::2.0.1.2 (::2.0.1.2)  1.994 ms  2.301 ms  2.016 ms
 3  ::2.0.2.1 (::2.0.2.1)  18.846 ms  30.582 ms  19.776 ms
 4  2000:0:0:40::2 (2000:0:0:40::2)  48.964 ms  60.131 ms  53.895 ms
```

## 9.  Decapsulating Behavior

If this uniform ttl model is enabled on the decapsulating NVE, and the overlay header indicates that uniform ttl model applies (the T-bit in the case of VXLAN), then the NVE will perform these additional operations as part of decapsulating a packet where the inner packet is an IPv4 or IPv6 packet:

o  Examine the outer IPv4 TTL (or outer IPv6 hopcount, respectively) on receipt and if 1 or less, then drop the packet and send an outer ICMPv4 (or ICMPv6) ttl exceeded back to the source of the outer packet i.e., the ingress NVE. This ICMP packet should look the same as an ICMP error generated by an underlay router, and the requirement in [RFC1812] on the size of the packet in error applies.

- o If ttl did not expire, then decrement the above ttl/hopcount and place it in the inner IP header. If the inner IP header is IPv4 then update the IPv4 header checksum. Then decapsulate and send the packet as for other decapsulated packets.

- o If some other errors prevent sending the packet (such as unknown VN Context Id), then the NVE SHOULD send an ICMP host unreachable instead of a ttl exceeded error.

## 10. Other ICMP errors

The technique for selecting ttl behavior specified in this draft can also be used to trigger other ICMPv4 and ICMPv6 errors. For example, [RFC1933] specifies how ICMP packet too big from underlay routers can be used to report over ICMP packet too big errors to the original source. Other errors that are more specific to the overlay protocol might also be useful, such as not being able to find a VNI ID for the incoming port,vlan, or not being able to flood the packet if the packet is a Broadcast, Unknown unicast, or Multicast packet.

## 11. Downstream Egress Paths Object

The Downstream Egress Paths Object MAY be appended to the ICMP Time Exceeded and Destination Unreachable messages. A single instance of the Downstream Egress Paths Object represents the egress paths at the router that sends the ICMP message. The Downstream Egress Paths Object must be preceded by an ICMP Extension Structure Header and an ICMP Object Header. Both are defined in [RFC4884]. The format follows closely [RFC4379] with some generalizations for Multipath types.

Class-Num = TBA by IANA, Downstream Egress Paths Class

C-Type = 1.

If the replying router is the destination of the echo request, then a Downstream Egress Paths Object SHOULD NOT be included in the ICMP Error message. Otherwise the replying router MAY append a Downstream Egress Paths Object for all interfaces over which the echo request packet could be forwarded.

The Object Length is K*N + M*N, where M is the Multipath Length for each egress path, M may not be the same for different paths. Values for K are found in the description of Address Type below.

The Downstream Egress Paths Object has the following format:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Path-1        MTU             | Address Type  | Reserved      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Downstream IP Address (4 or 16 octets)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Downstream Interface Address (4 or 16 octets)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| MultipathType |       Multipath Length        | Reserved      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                     (Multipath Information)                   .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                                                               ~
~                                                               ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Path-N        MTU             | Address Type  | Reserved      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Downstream IP Address (4 or 16 octets)            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Downstream Interface Address (4 or 16 octets)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| MultipathType |      Multipath Length         | Reserved      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                     (Multipath Information)                   .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Downstream Egress Paths Object

Maximum Transmission Unit (MTU):
The MTU is the size in octets of the largest IP frame that fits on the downstream interface.

Address Type:
The Address Type indicates if the interface is numbered or unnumbered. It also determines the length of the Downstream IP Address and Downstream Interface fields. The resulting total for the initial part of the one path of the downstream Egress Paths

Object is listed in the table below as "K Octets".

The Address Type is set to one of the following values:

| Type # | Address Type | K Octets |
|---|---|---|
| 1 | IPv4 Numbered | 16 |
| 2 | IPv4 Unnumbered | 16 |
| 3 | IPv6 Numbered | 40 |
| 4 | IPv6 Unnumbered | 28 |

Downstream IP Address and Downstream Interface Address:
IPv4 addresses and interface indices are encoded in 4 octets; IPv6 addresses are encoded in 16 octets.

If the interface to the downstream router has a unique IP address (e.g., it is numbered and not a LAG), then the Address Type MUST be set to IPv4 or IPv6, the Downstream IP Address MUST be set to either the downstream router’s Router ID or the interface address of the downstream router, and the Downstream Interface Address MUST be set to the downstream router’s interface address.

If the interface to the downstream router does not have a unique IP address (e.g., it is is unnumbered or a LAG), the Address Type MUST be IPv4 Unnumbered or IPv6 Unnumbered, the Downstream IP Address MUST be the downstream router’s Router ID or the interface address of the downstream router, and the Downstream Interface Address MUST be set to the index assigned by the upstream router to the interface.

Multipath Type:
The following Multipath Types are defined:

| Key | Type | Multipath Information |
|---|---|---|
| 0 | no multipath | Empty (Multipath Length = 0) |
| 1 | MAC SA/DA | Inner MAC in tunnel payload |
| 2 | IP Src/Dest | Inner IP src/dest in tunnel payload |
| 3 | L4 src port | L4 src ports in tunnel payload |
| 4 | L4 src port range | low/high L4 src port pairs |

Type 0 indicates that all packets will be forwarded out this one interface.

Types 1 through 4 specify that the supplied Multipath Information will serve to exercise this path.

Multipath Length:
The length in octets of the Multipath Information.

Multipath Information:
The Multipath Information encodes L4 source ports that will exercise this path. The Multipath Information depends on the Multipath Type. The contents of the field are shown in the table above. For Type 4, ranges indicated by L4 source port pairs MUST NOT overlap and MUST be in ascending sequence.

## 12. Security Considerations

The considerations in [I-D.ietf-nvo3-security-requirements] apply.

In addition, the use of the uniform ttl tunnel model will result in ICMP errors being generated by underlay routers and consumed by NVEs. That resents an attack vector which does not exist in a pipe ttl tunnel model. However, ICMP errors should be rate limited [RFC1812]. Implementations should also take appropriate measures in rate limiting the input rate for ICMP errors that are processed by limited CPU resources.

Some implementations might handle the trace packets (with uniform ttl model) in software while the pipe ttl model packets can be handled in hardware. In such a case the implementation should have mechanisms to avoid starvation of limited CPU resources due to these packets.

## 13. IANA Considerations

TBD

## 14. Acknowledgements

The authors acknowledge the helpful comments from David Black and Diego Garcia del Rio.

## 15. References

### 15.1. Normative References

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <http://www.rfc-editor.org/info/rfc792>.

[RFC1812] Baker, F., Ed., "Requirements for IP Version 4 Routers", RFC 1812, DOI 10.17487/RFC1812, June 1995, <http://www.rfc-editor.org/info/rfc1812>.

[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>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <http://www.rfc-editor.org/info/rfc7348>.

[RFC7365] Lasserre, M., Balus, F., Morin, T., Bitar, N., and Y. Rekhter, "Framework for Data Center (DC) Network Virtualization", RFC 7365, DOI 10.17487/RFC7365, October 2014, <http://www.rfc-editor.org/info/rfc7365>.

### 15.2. Informative References

[I-D.gross-geneve]
Gross, J., Sridhar, T., Garg, P., Wright, C., Ganga, I., Agarwal, P., Duda, K., Dutt, D., and J. Hudson, "Geneve: Generic Network Virtualization Encapsulation", draft-gross-geneve-02 (work in progress), October 2014.

[I-D.herbert-gue]
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-herbert-gue-03 (work in progress), March 2015.

[I-D.ietf-nvo3-security-requirements]
Hartman, S., Zhang, D., Wasserman, M., Qiang, Z., and M. Zhang, "Security Requirements of NVO3", draft-ietf-nvo3-security-requirements-07 (work in progress), June 2016.

[I-D.sridharan-virtualization-nvgre]
Garg, P. and Y. Wang, "NVGRE: Network Virtualization using

Generic Routing Encapsulation",
draft-sridharan-virtualization-nvgre-08 (work in progress), April 2015.

[I-D.tissa-lime-yang-oam-model]
Senevirathne, T., Finn, N., Kumar, D., Salam, S., Wu, Q., and Z. Wang, "Generic YANG Data Model for Operations, Administration, and Maintenance (OAM)", draft-tissa-lime-yang-oam-model-06 (work in progress), August 2015.

[RFC1933] Gilligan, R. and E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers", RFC 1933, DOI 10.17487/RFC1933, April 1996, <http://www.rfc-editor.org/info/rfc1933>.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <http://www.rfc-editor.org/info/rfc2473>.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <http://www.rfc-editor.org/info/rfc2983>.

[RFC3270] Le Faucheur, F., Wu, L., Davie, B., Davari, S., Vaananen, P., Krishnan, R., Cheval, P., and J. Heinanen, "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services", RFC 3270, DOI 10.17487/RFC3270, May 2002, <http://www.rfc-editor.org/info/rfc3270>.

[RFC3443] Agarwal, P. and B. Akyol, "Time To Live (TTL) Processing in Multi-Protocol Label Switching (MPLS) Networks", RFC 3443, DOI 10.17487/RFC3443, January 2003, <http://www.rfc-editor.org/info/rfc3443>.

[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>.

[RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, DOI 10.17487/RFC4884, April 2007, <http://www.rfc-editor.org/info/rfc4884>.

[RFC4950] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "ICMP Extensions for Multiprotocol Label Switching", RFC 4950, DOI 10.17487/RFC4950, August 2007, <http://www.rfc-editor.org/info/rfc4950>.

Authors' Addresses

Erik Nordmark
Arista Networks
Santa Clara, CA
USA

Email: nordmark@arista.com

Chandra Appanna
Arista Networks
Santa Clara, CA
USA

Email: achandra@arista.com

Alton Lo
Arista Networks
Santa Clara, CA
USA

Email: altonlo@arista.com

Sami Boutros
VMware

Email: sboutros@vmware.com

Ankur Dubey
VMware

Email: adubey@vmware.com

Network Working Group T. Taylor, Ed.
Internet-Draft PT Taylor Consulting
Intended status: Informational Y. Zhuang, Ed.
Expires: June 13, 2016 Huawei
December 11, 2015

# Applicability of Generic YANG Data Model for layer Independent OAM Management
draft-zhuang-lime-yang-oam-model-applicability-02

## Abstract

A generic YANG data model for Operations, Administration, and Maintenance (OAM) has been defined in [GENYANGOAM], with the intention that technology-specific extensions will be developed to be able reference/use the Generic YANG model. In this document, we describe the applicability of the generic YANG OAM data model to specific OAM technologies. To be concrete, we also demonstrate the usability and extensibility of the generic YANG OAM model with OAM protocols such as IP Ping, traceroute, BFD and MPLS LSP Ping.

## Status of This Memo



## Copyright Notice

Table of Contents

## 1. Introduction

The Generic YANG [RFC6020] over NETCONF [RFC6241] data model for OAM defined in [GENYANGOAM], aims at providing consistent configuration, reporting and representation of OAM mechanisms at any layer for any technology.

In this document, we discuss the applicability of the generic YANG OAM model to various OAM technologies and demonstrates that the YANG model(s) developed in the LIME WG are usable and extensible for those technologies. The demonstration uses IP Ping, traceroute, BFD and LSP Ping as specific examples.

## 2. Conventions Used In This Document

This document contains no normative language.

### 2.1. Terminology

MP Maintenance Point [IEEE802.1Q].

MEP  Maintenance association End Point [RFC7174] [IEEE802.1Q] [RFC6371].

MIP  Maintenance domain Intermediate Point [RFC7174] [IEEE802.1Q] [RFC6371].

MA Maintenance Association [IEEE802.1Q] [RFC7174].

MD Maintenance Domain [IEEE802.1Q].

OAM  Operations, Administration, and Maintenance [RFC6291].

TRILL  Transparent Interconnection of Lots of Links [RFC6325].

RPC  Remote Procedure Call[RFC6020].

## 3.  Basic Structure of Generic YANG Model for OAM

As the basis of this document, the generic YANG model for OAM specified as the LIME base model is shown in Figure 1.

```
module: ietf-gen-oam
      +--rw domains
         +--rw domain* [technology MD-name-string]
            +--rw technology        identityref
            +--rw MD-name-string    MD-name-string
            ...
            +--rw MAs
               +--rw MA* [MA-name-string]
                  +--rw MA-name-string       MA-name-string
                  ...
                  +--rw MEP* [mep-name]
                  |  +--rw mep-name             MEP-name
                  |  ...
                  |  +--rw session* [session-cookie]
                  |  ...
                  +--rw MIP* [interface]
                  |  +--rw interface    if:interface-ref
                  +--rw related-oam-layer* [offset]
                     ...
   rpcs:
      +---x continuity-check
      |     ...
      +---x continuity-verification     {connectivity-verification}?
      |     ...
      +---x path-discovery
         ...
   notifications:
      +---n defect-condition-notification
         ...
```

Figure 1: Structure of the Generic LIME Base Model

The generic YANG OAM model comprises three definitions for configuration and operational state data:

o  configuration model definition;

o  Remote procedure call (RPC) definition;

o  and notification definition.

The configuration model definition provides hierarchical structure to describe fault domain (i.e., maintenance domain), test point (i.e., maintenance point), technology type, layering, and session context for trouble-shooting.  This basic configuration model enables users to select corresponding layers and nodes serving as anchor points to define their specific technology OAM YANG models.

The RPC definition provides uniform APIs for common OAM functions such as continuity check, connectivity verification, path discovery, performance measurement and their equivalents.  These APIs are used by the network management system (NMS) to control OAM tools and functionalities on network elements for measuring and monitoring the data plane (e.g., LSP Ping, IP performance measurement protocol) and troubleshooting (e.g., fault localization).  These OAM tools activation can be pro-active and on-demand.

The notification definition also provides a uniform API to report defects, faults, and network failures at different layers.  This API is used by network elements to report to the network management system (NMS).  The content of each notification includes the fault domain and the test point(s) that detected the fault and may generate the error message.  This API must be activated proactively.

## 3.1.  Performance Management Support

To support OAM Performance Management, the generic YANG Data Model for OAM needs to be extended by adding loss and delay measurements support with the following model structure:

```
/* MEP Configuration extension */
 augment /goam:domains/goam:domain/goam:MAs/goam:MA/goam:MEP:
    +--rw delay-measurements?
  augment /goam:domains/goam:domain/goam:MAs/goam:MA/goam:MEP:
    +--rw loss-measurements?
/* New rpcs */
rpcs:
  +---x create-loss-measurement
  |     ...
  +---x abort-loss-measurement
  |     ...
  +---x create-delay-measurement
  |     ...
  +---x abort-delay-measurement
  |     ...
```

Both pro-active and on-demand loss and delay measurement are supported by augument MEP configuration and RPCs with session type parameter.  The details of Performance management extension is specified in the [I-D.wang-lime-yang-pm]

# 4.  Guidelines For Extending the LIME Base Data Model

YANG allows a module to reference external modules to reuse data already defined in those modules.  Therefore a technology-specific model can import data definitions from the LIME base model.

The import statements are used to make definitions available inside other modules [RFC6020]. Users who want to develop a technology-specific OAM model should import the ietf-gen-oam YANG model with the following statements:

```
module example-ietf-xxx-oam {
        namespace "urn:foo:params:xml:ns:yang:ietf-xxx-oam";
        prefix xxxoam;

        import ietf-gen-oam {
          prefix goam;
        }
        ......
```

As described in Section 3, the LIME base model provides a hierarchical structure for configuration, notification and RPCs. Each of these three aspects should be extended with technology-specific features and parameters relating to each technology of interest.

YANG allows a module to insert additional nodes into data models, including both the current module (and its submodules) or an external module. This is useful to let specific technologies add specific parameters into the LIME base model.

Here we summarize four ways to extend the LIME base model for specific technologies:

- Extend structure for configuration with technology specific parameters

- Extend structure for notification with technology specific parameters

- Extend structure for RPC with technology specific parameters

- Define new RPCs and notifications in the technology specific OAM data model.

### 4.1. Extend configuration structure with technology specific parameters

As described in [RFC6020], the "augment" statement defines the location in the data model hierarchy where new nodes are inserted.

By using the "augment" statement, the hierarchy of configuration structure can be extended with new data nodes that express technology-specific parameters to meet the requirements of the respective technologies. The technology-specific model developer must take care to select the right layers and nodes in the configuration structure as anchor points to insert these additional data.

For example, assume a technology- specific OAM YANG model A.  An "a" node needs to be inserted within the MA (Maintenance Association):

```
augment /goam:domains/goam:domain/goam:MAs/goam:MA:
          +--a?   foo
```

Corresponding YANG encoding:

```
augment "/goam:domains/goam:domain/goam:MAs/goam:MA"{
         leaf a{
          type foo
           description
           "foo";
 }
}
```

There are the following five levels in the hierarchy of configuration structure which we can choose as anchor point to insert additional data definitions:

- o  Maintenance domain (MD) at the root level;
- o  Maintenance Association (MA) at the second level;
- o  Maintenance Association Endpoint (MEP) and Maintenance Association Intermediate point(MIP) at the third level;
- o  Session at the fourth level;
- o  Interface at the fifth level;

#### 4.1.1.  Maintenance domain (MD) at the root level

At the Maintenance Domain level, domain data node at root level can be augmented with technology type.  [GENYANGOAM] defines a new globally unique, abstract, and untyped "technology-types" base identity by using the "identity" statement. "identity" and "identityref" are used to Identify New Technology Types.  Each technology-specific module then can extend technology type in the base model and specifies a corresponding concrete identity using this base: ipv4, ipv6, trill, mpls, etc.

### 4.1.2. Maintenance Association (MA) at the second level

At the Maintenance Association level, an MA data node can be augmented with connectivity context information. For example:

```
+--rw MAs
    +--rw MA* [MA-name-string]
       ...
       +--rw (connectivity-context)?
       |  +--:(context-null)
       |     +--rw context-null?        Empty
```

Corresponding YANG encoding:

```
choice connectivity-context {
  default "context-null";
  case context-null {
    description
      "this is a place holder when no context is needed";
    leaf context-null {
      type empty;
      description
        "there is no context defined";
    }
  }
  description
    "connectivity context";
}
```

ietf-gen-oam YANG model users who want to define a specific OAM technology model can augment the corresponding choice node by defining a new case to carry technology specific extensions.

For example, for a specific OAM technology YANG model A, an "a" node is needed to indicate the connectivity context for this specific OAM technology. To achieve this, it is only necessary to augment the connectivity-context choice node in the ietf-gen-OAM YANG model by defining a "connectivity-context-A" case as:

```
augment /goam:domains/goam:domain/goam:MAs/goam:MA
/goam:connectivity-context:
      +--:(connectivity-context-A)
         +--a?   foo
```

Corresponding YANG encoding:

```
augment "/goam:domains/goam:domain/goam:MAs/goam:MA"
+"/goam:connectivity-context" {
   case connectivity-context-A {
    leaf a{
     type foo;}
    }
   }
```

In some case when technology type in the Maintenance Domain level is not sufficient to identify OAM technology with different encapsulation method, MA data node can be further augmented with technology sub type (see an example in the section 5.5).

### 4.1.3. Maintenance Association Endpoint (MEP) at the third level

At the Maintenance Association Endpoint level, a MEP data node can be augmented with connectivity-context information, ECMP information and session information respectively.

### 4.1.4. Session at the fourth level

At the session level, Session data node can be augmented with technology specific information such as Session type, Session interval,etc.

### 4.1.5. Interface at the fifth level

At the interface level under MEP/MIP or under session, the interface data node can be augmented with technology specific information such as context information, interface type,disable/enable button,etc.

## 4.2. Extend RPC structure with technology specific parameters

[GENYANGOAM] defines rpc model which abstracts OAM specific commands in a technology independent manner. In this RPC model, three generic RPC commands are specified. By using the "augment" statement,the RPC structure for each OAM command can be extended with new data nodes that express technology-specific OAM command parameters to meet the requirements of the respective technologies. The technology-specific model developer must take care to select the right layers and nodes in the RPC structure as anchor points to insert these additional data.  There are two places which we can choose as anchor point to insert additional data definitions:

o  Input data node

Input data node can be augmented with technology type, sub-command type, session type and other technology specific parameters.  Here is an example of sub-command type:

[GENYANGOAM] defines a "command-sub-type" abstract identity for different RPC commands, e.g., to distinguish the types of IP ping [RFC792], LSP ping [RFC4379].  Use of this identity is optional for most cases.

The corresponding statements are shown as below.

```
identity command-sub-type {
  description
    "defines different rpc command subtypes, e.g rfc792 IP
     ping, rfc4379 LSP ping, this is
     optional for most cases";
}

identity icmp-rfc792 {
  base command-sub-type;
  description
    "Defines the command subtypes for ICMPv4 ping";
  reference "RFC 792";
}

identity icmp-rfc4443 {
  base command-sub-type;
  description
    "Defines the command subtypes for ICMPv6 ping";
  reference "RFC 4443";
}

identity icmp-rfc4379 {
  base command-sub-type;
  description
    "Defines the command subtypes for LSP ping";
  reference "RFC 4379";
}
```

o  Output data node

Similarly, output data nod can be augmented with technology specific test results information collected by executing OAM command.

### 4.3. Extend Notification structure with technology specific parameters

[GENYANGOAM] defines one notification model which abstracts defects notification in a technology independent manner. By using the "augment" statement,the notification structure can be extended with new data nodes that express technology-specific notification parameters to meet the requirements of the respective technologies. The technology-specific model developer must take care to select the right layers and nodes in the notification structure as anchor points to insert these additional data.

### 4.4. Define New RPCs and Notifications

The LIME base model presents three basic RPCs: continuity check, connectivity verification and path discovery. Technology-specific OAM models can either extend the existing RPCs and notifications defined in the LIME base model or define new RPCs and notifications if generic RPCs and notifications cannot be reused to meet their requirements.

For example, a Multicast Tree Verification (MTV) [TRILLOAMYANG] RPC command is defined in the TRILL OAM model to verify connectivity as well as data-plane and control-plane integrity of TRILL multicast forwarding as follows:

```
RPCs:
   +---x mtv
      +--ro input
      |  +--ro technology              identityref
      |  +--ro MD-name-string          MD-name-string
      |  +--ro MA-name-string?         MA-name-string
      |  ...
      +--ro output
         +--ro response* [mep-address mep-id]
            +--ro hop-count?                   uint8
            +--ro mep-id                       tril-rb-nickname
            +--ro mep-address                  tril-rb-nickname
            ...
```

## 5. Applicability of LIME Model to Various Technologies

As mentioned above, the ietf-gen-oam model describes the abstract common core configuration, statistics, RPCs, and notifications for layer independent OAM management.

Following guidelines stated in Section 4, ietf-gen-oam YANG model users can augment this base model by defining and adding new data nodes with technology specific functions and parameters into proper anchor points of the ietf-gen-oam model, so as to develop a technology-specific OAM model.

With these guidelines in hand, this section further demonstrates the usability of the ietf-gen-oam YANG model to various OAM technologies. Note that, in this section, we only present several snippets of technology-specific data model extensions for illustrative purposes. The complete model extensions should be worked on in respective protocol working groups.

## 5.1. Generic YANG Model extension for IP OAM

### 5.1.1. MD Configuration Extension

MD level configuration parameters are management information which can be inherited in the TRILL OAM model and set by LIME base model as default values. For example domain name can be set to area-ID in the IP OAM case. In addition, at the Maintenance Domain level, domain data node at root level can be augmented with technology type.

Note that MD level configuration parameters provides context information for management system to correlate faults, defects, network failures with location information, which helps quickly identify root causes of network failures.MD level configuration parameters MUST not be carried using IP Ping and traceroute protocol since IP Ping and traceroute doesn’t support transport of these management information.

#### 5.1.1.1. Technology Type Extension

The technology types ipv4 and ipv6 have already been defined in the LIME base model. Therefore no technology type extension is required in the IP OAM model.

### 5.1.2. MA Configuration Extension

MA level configuration parameters are management information which can be inherited in the IP OAM model and set by LIME base model as default values. In addition, at the Maintenance Association(MA) level, MA data node at the second level can be augmented with connectivity-context extension.

Note that MA level configuration parameters provides context information for management system to correlate faults, defects, network failures with location information, which helps quickly identify root causes of network failures.MA level configuration parameters MUST not be carried using IP Ping and traceroute protocol since IP Ping and traceroute doesn’t support transport of these management information.

##### 5.1.2.1. Connectivity-Context Extension

In IP OAM, one example of the connectivity-context is a 12 bit VLAN ID. The LIME base model defines a placeholder for connectivity-context. This allows other technologies to easily augment it to include technology specific extensions. The snippet below depicts an example of augmenting context-id to include VLAN ID.

```
augment /goam:domains/goam:domain/goam:MAs/goam:MA
/goam:MEP/goam:connectivity-context:
   +--:(context-id-vlan)
      +--rw context-id-vlan?   vlan
augment /goam:domains/goam:domain/goam:MAs/goam:MA/goam:MEP
/goam:session/goam:connectivity-context:
   +--:(context-id-vlan)
      +--rw context-id-vlan?   vlan
```

#### 5.1.3. MEP Configuration Extension

MEP configuration in the LIME base model already supports configuring the interface on which the MEP is located with an IP address. There is no additional MEP configuration extension needed for IP OAM.

However, IP Ping, traceroute do not use the MEPID in their message headers. Therefore it is important to have method to derive the MEPID in an automatic manner with no user intervention.

##### 5.1.3.1. ECMP extension

The flow-entropy parameter in the LIME OAM configuration model is an optional parameter. Since standard IP OAM protocols, e.g., IP Ping and Traceroute, don’t support ECMP path selection, the flow-entropy parameter does not need to be supported in the IP OAM model.

#### 5.1.4. RPC Extension

Technology type in the RPC definition has already been defined in the LIME OAM base model. Therefore no technology type extension is required in the RPC definition. For IP OAM, IP Ping and IP Traceroute RPCs need to be supported. For the IP OAM model, the continuity-check RPC with IPv4 or IPv6 as technology type can be mapped to the IP Ping RPC, while the path-discovery RPC with IPv4 or IPv6 as technology type can be mapped to IP Traceroute.

#### 5.1.5. Performance Monitoring Extension

Editor Note: IP performance measurement (IPPM) and IP Ping and Traceroute are discussed separately based on the [RFC7276] classification of OAM technologies. Although IPPM and IP OAM are both applied to the IP network, based on Table 4 of [RFC7276], IP OAM does not support performance measurement. It is necessary to use OWAMP and TWAMP, defined in IPPM, for that purpose.

##### 5.1.5.1. MEP PM Configuration Extension

To support IP performance measurement, MEP configuration in the LIME base model can be extended with:

- o loss-stats-group: grouping object for loss measurement session statistics.
- o measurement-timing-group: grouping object used for proactive and on-demand scheduling of PM measurement sessions.
- o delay-measurement-configuration-group: grouping configuration object for the delay measurement function.
- o delay-measurement-stats-group: grouping object for delay measurement session statistics.
- o loss-measurement-configuration-group: grouping configuration object for the loss measurement function.
- o loss-measurement-stats-group: grouping object for loss measurement session statistics.

##### 5.1.5.2. RPC PM Extension

To support IP performance measurement, it is recommended that four RPCs are defined in the IPPM model:

- o create-loss-measurement RPC: allows scheduling of one-way or two-way on-demand or proactive performance monitoring loss measurement sessions.
- o abort-loss-measurement RPC: allows aborting of currently running or scheduled loss measurement session.
- o create-delay-measurement RPC: allows scheduling of one-way or two-way on-demand or proactive performance monitoring delay measurement sessions.

o abort-delay-measurement RPC: allows aborting of currently running or scheduled delay measurement sessions.

## 5.2. Generic YANG Model extension for TRILL OAM

### 5.2.1. MD Configuration Extension

MD level configuration parameters are management information which can be inherited in the TRILL OAM model and set by LIME base model as default values. For example domain name can be set to area-ID in the TRILL OAM case. In addition, at the Maintenance Domain level, domain data node at root level can be augmented with technology type.

Note that MD level configuration parameters provides context information for management system to correlate faults, defects, network failures with location information, which helps quickly identify root causes of network failures.

#### 5.2.1.1. Technology Type Extension

No TRILL technology type has been defined in the LIME base model. Therefore a technology type extension is required in the TRILL OAM model. The technology type "trill" is defined as an identity that augments the base "technology-types" defined in the LIME base model:

```
identity trill{
 base goam:technology-types;
 description
  "trill type";
}
```

### 5.2.2. MA Configuration Extension

MA level configuration parameters are management information which can be inherited in the TRILL OAM model and set by LIME base model as default values. In addition, at the Maintenance Association(MA) level, MA data node at the second level can be augmented with connectivity-context extension.

Note that MA level configuration parameters provides context information for management system to correlate faults, defects, network failures with location information, which helps quickly identify root causes of network failures.

#### 5.2.2.1. Connectivity-Context Extension

In TRILL OAM, one example of connectivity-context is either a 12 bit VLAN ID or a 24 bit Fine Grain Label. The LIME base model defines a placeholder for context-id. This allows other technologies to easily augment that to include technology specific extensions. The snippet below depicts an example of augmenting connectivity-context to include either VLAN ID or Fine Grain Label.

```
augment /goam:domains/goam:domain/goam:MAs
/goam:MA /goam:connectivity-context:
      +--:(connectivity-context-vlan)
      |  +--rw connectivity-context-vlan?   vlan
      +--:(connectivity-context-fgl)
         +--rw connectivity-context-fgl?    fgl

augment /goam:domains/goam:domain/goam:MAs/goam:MA/goam:MEP
/goam:session/goam:connectivity-context:
      +--:(connectivity-context-vlan)
      |  +--rw connectivity-context-vlan?   vlan
      +--:(connectivity-context-fgl)
         +--rw connectivity-context-fgl?    fgl
```

### 5.2.3. MEP Configuration Extension

The MEP configuration definition in the LIME base model already supports configuring the interface of MEP with either MAC address or IP address. In addition, the MEP address can be represented using a 2 octet RBridge Nickname in TRILL OAM . Hence, the TRILL OAM model augments the MEP configuration in base model to add a nickname case into the MEP address choice node as follows:

```
augment /goam:domains/goam:domain/goam:MAs
/goam:MA/ goam:MEP/goam:mep-address:
      +--:( mep-address-trill)
      |  +--rw mep-address-trill?  tril-rb-nickname
```

In addition, at the Maintenance Association Endpoint(MEP) level, MEP data node at the third level can be augmented with ECMP extension.

#### 5.2.3.1. ECMP Extension

The flow-entropy parameter in the LIME base model is an optional parameter. Since TRILL supports ECMP path selection, flow-entropy in TRILL is defined as a 96 octet field. The snippet below illustrates its extension.

```
augment /goam:domains/goam:domain/goam:MAs/goam:MA/goam:MEP
/goam:flow-entropy:
      +--:(flow-entropy-trill)
         +--rw flow-entropy-trill?   flow-entropy-trill
augment /goam:domains/goam:domain/goam:MAs/goam:MA/goam:MEP
/goam:session/goam:flow-entropy:
      +--:(flow-entropy-trill)
         +--rw flow-entropy-trill?   flow-entropy-trill
```

### 5.2.4. RPC Extension

In the TRILL OAM YANG model, the continuity-check and path-discovery RPC commands are extended with TRILL specific requirements. The snippet below illustrates the TRILL OAM RPC extension.

```
augment /goam:continuity-check/goam:input:
      +--ro (out-of-band)?
      |  +--:(ipv4-address)
      |  |  +--ro ipv4-address?      inet:ipv4-address
      |  +--:(ipv6-address)
      |  |  +--ro ipv6-address?      inet:ipv6-address
      |  +--:(trill-nickname)
      |     +--ro trill-nickname?    tril-rb-nickname
      +--ro diagnostic-vlan?   boolean
augment /goam:continuity-check/goam:input/goam:flow-entropy:
      +--:(flow-entropy-trill)
         +--ro flow-entropy-trill?   flow-entropy-trill
augment /goam:continuity-check/goam:output:
      +--ro upstream-rbridge?   tril-rb-nickname
      +--ro next-hop-rbridge*   tril-rb-nickname
augment /goam:path-discovery/goam:input:
      +--ro (out-of-band)?
      |  +--:(ipv4-address)
      |  |  +--ro ipv4-address?      inet:ipv4-address
      |  +--:(ipv6-address)
      |  |  +--ro ipv6-address?      inet:ipv6-address
      |  +--:(trill-nickname)
      |     +--ro trill-nickname?    tril-rb-nickname
      +--ro diagnostic-vlan?   boolean
augment /goam:path-discovery/goam:input/goam:flow-entropy:
      +--:(flow-entropy-trill)
         +--ro flow-entropy-trill?   flow-entropy-trill
augment /goam:path-discovery/goam:output/goam:response:
      +--ro upstream-rbridge?   tril-rb-nickname
      +--ro next-hop-rbridge*   tril-rb-nickname
```

#### 5.2.5. Performance Management (PM) Extension

##### 5.2.5.1. MEP PM Configuration Extension

To support performance measurement for TRILL, MEP configuration in the LIME base model can be extended with:

- o loss-stats-group: grouping statistics object for TRILL Loss measurement sessions;
- o measurement-timing-group: grouping object used for proactive and on-demand scheduling of PM measurement sessions;
- o delay-measurement-configuration-group: grouping configuration object for TRILL delay measurement function;
- o delay-measurement-stats-group: grouping statistics object for TRILL delay measurement sessions.

##### 5.2.5.2. RPC PM Extension

To support performance measurement for TRILL, it is recommended that four new RPCs are defined in the TRILL OAM PM model:

- o create-loss-measurement RPC: allows scheduling of one-way or two-way on-demand or proactive performance monitoring loss measurement sessions.
- o abort-loss-measurement RPC: allows aborting of currently running or scheduled loss measurement sessions.
- o create-delay-measurement RPC: allows scheduling of one-way or two-way on-demand or proactive performance monitoring delay measurement sessions.
- o abort-delay-measurement RPC: allows aborting of currently running or scheduled delay measurement sessions.

#### 5.2.6. Usage example

This part gives a simple example of implementing the TRILL OAM model onto network devices.

The scenario is shown in Figure 2, in which there are two companies: A and B. Both have departments in City 1 and City 2. Meanwhile, different departments within the same company should be able to communicate with each other. However, the communication services of these two companies should be separated from each other.

To meet the requirements above, two Ethernet Lease line, E-Line-1 and E-Line-2, are set between NE1 and NE3: to isolate the communication traffic between two companies. VLAN 100 associates port 3-EFF8-1 of NE1 facing with company A while VLAN 200 associates port 3-EF8-2 of NE1 facing with company B. For network maintenance, NE1, NE2 and NE3 are within a same maintenance domain: MD1. Two maintenance associations MA1 and MA2 are configured and stand for E-Line-1 and E-Line-2 under MD1. The MAC addresses of NE1, NE2, NE3 are MAC-FOO1, MAC-FOO2, MAC-FOO3 respectively.

```
                +------+       +-----+       +------+
                |      |       |     |       |      |
                |  NE1 +-------| NE2 |-------+  NE3 |
                |      |       |     |       |      |
        +-------+--+---+       +-----+       +---+--+---------+
3-EFF8-1|          |3-EFF8-2                     |            |
        |          |                             |            |
      +-+-+     +--++                          +-+-+        +-+-+
      |   |     |   |                          |   |        |   |
      +---+     +---+                          +---+        +---+
       A          B                              A            B
           CITY1                                      CITY2
```

Figure 2: TRILL OAM scenario

#### 5.2.6.1. TRILL OAM Extension

To fulfill the TRILL OAM configuration, the LME base model should be extended by augmenting the connectivity-context and inserting a port node in the MEP list. The snippet below illustrates an example of TRILL OAM model extension.

```
augment /goam:domains/goam:domain/goam:MAs
/goam:MA/goam:MEP /goam:mep-address:
      +--:( mep-address-trill)
      |  +--rw mep-address-trill?  tril-rb-nickname
augment /goam:domains/goam:domain/goam:MAs/goam:MA
/goam:connectivity-context:
      +--:(connectivity-context-vlan)
      |  +--rw connectivity-context-vlan?   vlan
      +--:(connectivity-context-fgl)
         +--rw connectivity-context-fgl?    fgl

augment /goam:domains/goam:domain/goam:MAs/goam:MA/goam:MEP
/goam:session/goam:connectivity-context:
      +--:(connectivity-context-vlan)
      |  +--rw connectivity-context-vlan?   vlan
      +--:(connectivity-context-fgl)
```

```
            +--rw connectivity-context-fgl?    fgl
   augment /goam:domains/goam:domain/goam:MAs/goam:MA
   /goam:MEP/goam:flow-entropy:
         +--:(flow-entropy-trill)
            +--rw flow-entropy-trill?   flow-entropy-trill
   augment /goam:domains/goam:domain/goam:MAs/goam:MA/goam:MEP
   /goam:session/goam:flow-entropy:
         +--:(flow-entropy-trill)
            +--rw flow-entropy-trill?   flow-entropy-trill
   augment /goam:continuity-check/goam:input:
         +--ro (out-of-band)?
         |  +--:(ipv4-address)
         |  |  +--ro ipv4-address?      inet:ipv4-address
         |  +--:(ipv6-address)
         |  |  +--ro ipv6-address?      inet:ipv6-address
         |  +--:(trill-nickname)
         |     +--ro trill-nickname?    tril-rb-nickname
         +--ro diagnostic-vlan?   boolean
   augment /goam:continuity-check/goam:input/goam:flow-entropy:
         +--:(flow-entropy-trill)
            +--ro flow-entropy-trill?   flow-entropy-trill
   augment /goam:continuity-check/goam:output:
         +--ro upstream-rbridge?   tril-rb-nickname
         +--ro next-hop-rbridge*   tril-rb-nickname
   augment /goam:path-discovery/goam:input:
         +--ro (out-of-band)?
         |  +--:(ipv4-address)
         |  |  +--ro ipv4-address?      inet:ipv4-address
         |  +--:(ipv6-address)
         |  |  +--ro ipv6-address?      inet:ipv6-address
         |  +--:(trill-nickname)
         |     +--ro trill-nickname?    tril-rb-nickname
         +--ro diagnostic-vlan?   boolean
   augment /goam:path-discovery/goam:input/goam:flow-entropy:
         +--:(flow-entropy-trill)
            +--ro flow-entropy-trill?   flow-entropy-trill
   augment /goam:path-discovery/goam:output/goam:response:
         +--ro upstream-rbridge?   tril-rb-nickname
         +--ro next-hop-rbridge*   tril-rb-nickname
```

#### 5.2.6.2. Corresponding XML Instance Example

This section gives an example of the corresponding XML instance for devices to implement the example TRILL OAM data models in Section 5.2.6.1.


```
      <domains>
       <domains>
```

```
        <technology> ethernet </techonlogy>
        <MD-name-string> MD1 </MD-name-string>
        <MAs>
         <MA>
          <MA-name-string>MA1</MA-name-string>
          <connectivity-context>
           <connectivity-context-vlan>
            100
           </connectivity-context-vlan>
          </connectivity-context>
          <MEP>
           <mep-name>NE1</mep-name>
           <mp-address>
            <mac-address>
             00-1E-4C-84-22-F1
            </mac-address>
           </mp-address>
          </MEP>
          <MEP>

           <mep-name>NE3</mep-name>
           <port>3-EFF8-1</port>
           <mp-address>
            <mac-address>
             00-1E-4C-84-22-F3
            </mac-address>
           </mp-address>
          </MEP>
          <MIP>NE2</MIP>
         </MA>
         <MA>
          <MA-name-string>MA2</MA-name-string>
          <connectivity-context>
           <connectivity-context-vlan>
            200
           </connectivity-context-vlan>
          </connectivity-context>
          <MEP>
           <mep-name>NE1</mep-name>
           <mp-address>
            <mac-address>
             00-1E-4C-84-22-F1
            </mac-address>
           </mp-address>
          </MEP>
          <MEP>
           <mep-name>NE3</mep-name>
           <mp-address>
```

```
            <mac-address>
             00-1E-4C-84-22-F3
            </mac-address>
           </mp-address>
          </MEP>
          <MIP>NE2</MIP>
         </MA>
        </MAs>
       </domains>
      </domains>
```

5.3. Generic YANG Model extension for MPLS OAM

5.3.1. MD Configuration Extension

MD level configuration parameters are management information which can be inherited in the MPLS OAM model and set by LIME base model as default values. For example domain name can be set to area-ID in the MPLS OAM case. In addition, at the Maintenance Domain level, domain data node at root level can be augmented with technology type and sub-technology type.

Note that MD level configuration parameters provides context information for management system to correlate faults, defects, network failures with location information, which helps quickly identify root causes of network failures. MD level configuration parameters MUST not be carried using MPLS OAM protocol(e.g., LSP Ping) since MPLS OAM protocol doesn’t support transport of these management information.

5.3.1.1. Technology Type Extension

No MPLS technology type has been defined in the LIME base model, hence it is required in the MPLS OAM model. The technology type "mpls" is defined as an identity that augments the base "technology-types" defined in the LIME base model:

```
identity mpls{
 base goam:technology-types;
 description
  "mpls type";
}
```

5.3.1.2. Sub Technology Type Extension

In MPLS, since different encapsulation types such as IP/UDP Encapsulation, PW-ACH encapsulation can be employed, the "technology-sub-type" data node is defined and added into the MPLS OAM model to further identify the encapsulation types within the MPLS OAM model. Based on it, we also define a technology sub-type for IP/UDP encapsulation and PW-ACH encapsulation.  Other Encapsulation types can be defined in the same way.

```
   identity technology-sub-type {
             description
               "certain implementations can have different
                encapsulation types such as ip/udp, pw-ach and so on.
                Instead of defining separate models for each
                encapsulation, we define a technology sub-type to
                further identify different encapsulations. Technology
                sub-type is associated at the MA level";
            }

        identity technology-sub-type-udp {
          base technology-sub-type;
          description
            "technology sub-type is IP/UDP encapsulation";
        }

        identity technology-sub-type-ach {
          base technology-sub-type;
          description
            "technology sub-type is PW-ACH encapsulation";
        }
        }

   augment "/goam:domains/goam:domain/goam:MAs/goam:MA" {
          leaf technology-sub-type {
            type identityref {
              base technology-sub-type;
            }
          }
        }
```

### 5.3.2.  MA Configuration Extension

MA level configuration parameters are management information which can be inherited in the MPLS- OAM model and set by LIME base model as default values.  In addition, at the Maintenance Association(MA) level, MA data node at the second level can be augmented with connectivity-context extension.

Note that MA level configuration parameters provides context information for management system to correlate faults, defects, network failures with location information, which helps quickly identify root causes of network failures.  MA level configuration parameters MUST not be carried using MPLS OAM protocol(e.g., LSP Ping) since MPLS OAM protocol doesn't support transport of these management information.

#### 5.3.2.1. Connectivity-Context Extension

In MPLS, one example of context-id is a 20 bit MPLS label. The LIME base model defines a placeholder for context-id. This allows other technologies to easily augment that to include technology specific extensions. The snippet below depicts an example of augmenting context-id to include per VRF MPLS labels in IP VPN or per CE MPLS labels in IP VPN.

```
augment "/goam:domains/goam:domain/goam:MAs/goam:MA
/goam:connectivity-context"
         {
           case connectivity-context-mpls {
              leaf vrf-label {
                type vrf-label;
              }
            }
         }
```

### 5.3.3. MEP Configuration Extension

In MPLS, the MEP address is either an IPv4 or IPV6 address in case IP/UDP encapsulation. MEP-ID is either a 2 octet unsigned integer value in case IP/UDP encapsulation or a variable length label value in case of G-ACH encapsulation. In the LIME base model, MEP-ID is defined as a variable length label value and the same definition can be used for MPLS with no further modification. In addition, at the Maintenance Association Endpoint(MEP) level, MEP data node at the third level can be augmented with Session extension and interface extension.

#### 5.3.3.1. ECMP Extension

Since MPLS supports ECMP path selection, the flow-entropy should be defined in MPLS OAM model. Technology type is used to extend the YANG model to specific usage.

```
   augment "/goam:domains/goam:domain/goam:MAs/goam:MA
   /goam:flow-entropy" {
          case flow-entropy-mpls {
            leaf flags-mpls {
              type flags-mpls;
            }
            leaf flow-entropy-mpls{
              type flow-entropy-mpls;
            }
          }
        }
```

5.3.3.2. Per interface Configuration Extension

TBC.

5.3.4. RPC Extension

5.3.4.1. CV extension for LSP Ping

5.3.4.2. Path Discovery Extension for LSP Ping

5.3.4.3. New RPC Alarm Indication Signal (AIS)

See [RFC6427].

5.3.4.4. New RPC for Lock Report (LKR)

See [RFC6427].

5.3.5. Performance Management Extension

5.3.5.1. MEP Configuration Extension

To support performance monitoring for MPLS, MEP configuration in the LIME base model can be extended with:

o TBC.

5.3.5.2. RPC Extension

To support performance monitoring for MPLS, it is recommended that five new RPCs are defined in the MPLS OAM PM model:

o MPLS Direct Loss Measurement (DLM) RPC [RFC6374];

o MPLS Inferred Loss Measurement (ILM) RPC [RFC6374];

- o  MPLS Delay Measurement (DM) RPC [RFC6374];
- o  MPLS Direct Loss and Delay Measurement RPC [RFC6374];
- o  MPLS Inferred Loss and Delay Measurement RPC [RFC6374].

#### 5.3.6.  Usage Example

In the MPLS tunnel scenario (see Figure 3): tunnel_1 is a static LSP tunnel passing through NE1-NE2-NE4.  It is used to perform LSP PING. tunnel_3 is another static LSP tunnel passing through NE4-NE2-NE1, used to bring back the LSP PING result.  tunnel_2 is a third static LSP tunnel passing through NE1-NE3-NE4, used to perform LSP Traceroute. tunnel_4 is a fourth static LSP tunnel passing through NE4-NE3-NE1, used to bring back the LSP Traceroute result.

```
                                 +-------+
                                 |       |
                +--------------->+  NE2  +----------------+
                | ...............|       |<.............. |
                | .              +-------+              . |
                | .                                     . |
                | v                                     . v
            +---+---+                                 +---+---+
            |       |                                 |       |
            |  NE1  |                                 |  NE4  |
            |       |                                 |       |
            +---+---+                                 +---+--^+
             .  ^                                         |  .
             .  |                                         |  .
             .  |                +-------+                |  .
             .  |                |       |                |  .
             .  +----------------+  NE3  +<---------------+  .
             ....................|       |....................
                                 +-------+


        ----- forward direction LSP tunnel
        ......backward direction LSP tunnel
```

Figure 3: MPLS OAM scenario

##### 5.3.6.1.  MPLS OAM Model Extension

TBD.

5.3.6.2. Corresponding XML Instance Example

TBD.

## 5.4. Generic YANG Model extension for MPLS-TP OAM

### 5.4.1. MD Configuration Extension

MD level configuration parameters are management information which can be inherited in the MPLS-TP OAM model and set by LIME base model as default values. For example domain name can be set to area-ID or the provider’s Autonomous System Number (ASN) [RFC6370] in the MPLS-TP OAM case. In addition, at the Maintenance Domain level, domain data node at root level can be augmented with technology type and sub-technology type.

Note that MD level configuration parameters provides context information for management system to correlate faults, defects, network failures with location information, which helps quickly identify root causes of network failures

5.4.1.1. Technology Type Extension

No MPLS-TP technology type has been defined in the LIME base model, hence it is required in the MPLS OAM model. The technology type "mpls-tp" is defined as an identity that augments the base "technology- types" defined in the LIME base model:

```
 identity mpls-tp{
       base goam:technology-types;
       description
        "mpls-tp type";
      }
```

5.4.1.2. Sub Technology Type Extension

In MPLS-TP, since different encapsulation types such as IP/UDP Encapsulation, PW-ACH encapsulation can be employed, the "technology-sub-type" data node is defined and added into the MPLS OAM model to further identify the encapsulation types within the MPLS-TP OAM model. Based on it, we also define a technology sub-type for IP/UDP encapsulation and PW-ACH encapsulation. Other Encapsulation types can be defined in the same way.

```
      identity technology-sub-type {
                description
                  "certain implementations can have different
                   encapsulation types such as ip/udp, pw-ach and so on.
                   Instead of defining separate models for each
                   encapsulation, we define a technology sub-type to
                   further identify different encapsulations. Technology
                   sub-type is associated at the MA level";
               }

           identity technology-sub-type-udp {
             base technology-sub-type;
             description
               "technology sub-type is IP/UDP encapsulation";
           }

           identity technology-sub-type-ach {
             base technology-sub-type;
             description
               "technology sub-type is PW-ACH encapsulation";
           }
           }

      augment "/goam:domains/goam:domain/goam:MAs/goam:MA" {
             leaf technology-sub-type {
               type identityref {
                 base technology-sub-type;
               }
             }
           }
```

### 5.4.2.  MA Configuration Extension

MA level configuration parameters are management information which can be inherited in the MPLS-TP OAM model and set by LIME base model as default values.  One example of MA Name is MEG LSP ID or MEG Section ID or MEG PW ID[RFC6370].  In addition, at the Maintenance Association(MA) level, MA data node at the second level can be augmented with connectivity-context extension.

Note that MA level configuration parameters provides context information for management system to correlate faults, defects, network failures with location information, which helps quickly identify root causes of network failures.

#### 5.4.2.1. Connectivity-Context Extension

In MPLS-TP, one example of context-id is a 20 bit MPLS label. The LIME base model defines a placeholder for context-id. This allows other technologies to easily augment that to include technology specific extensions. The snippet below depicts an example of augmenting context-id to include per VRF MPLS labels in IP VPN [RFC4364] or per CE MPLS labels in IP VPN [RFC4364].

```
augment "/goam:domains/goam:domain/goam:MAs/goam:MA
/goam:connectivity-context"
         {
           case connectivity-context-mpls {
              leaf vrf-label {
                type vrf-label;
              }
            }
         }
```

### 5.4.3. MEP Configuration Extension

In MPLS-TP, MEP-ID is either a variable length label value in case of G-ACH encapsulation or a 2 octet unsigned integer value in case of IP/UDP encapsulation. One example of MEP-ID is MPLS-TP LSP_MEP_ID [RFC6370]. In case of using IP/UDP encapsulation, the MEP address can be either an IPv4 or IPV6 address. In the LIME base model, MEP-ID is defined as a variable length label value and the same definition can be used for MPLS-TP with no further modification. In addition, at the Maintenance Association Endpoint(MEP) level, MEP data node at the third level can be augmented with Session extension and interface extension.

#### 5.4.3.1. ECMP Extension

The flow-entropy parameter in the LIME OAM configuration model is an optional parameter. Standard MPLS-TP OAM protocol does not support ECMP path selection, so the flow-entropy parameter does not need to be supported in the MPLS-TP OAM model.

#### 5.4.3.2. Per interface Configuration Extension

TBC.

### 5.4.4. RPC Extension

#### 5.4.4.1.  CC extension for MPLS-TP BFD CC Message

#### 5.4.4.2.  CV extension for MPLS-TP BFD CV Message

#### 5.4.4.3.  CV extension for On-Demand LSP CV with Non-IP Encapsulation

#### 5.4.4.4.  CV extension for On-Demand LSP CV with IP Encapsulation

#### 5.4.4.5.  New RPC for Remote Defect Indication

See [RFC6435].

#### 5.4.4.6.  New RPC for Lock Instruct

See [RFC6435].

### 5.4.5.  Performance Monitoring Extension

#### 5.4.5.1.  MEP Configuration Extension

To support performance monitoring for MPLS-TP, MEP configuration in the LIME base model can be extended with:

o  TBC.

#### 5.4.5.2.  RPC Extension

To support performance monitoring for MPLS-TP, it is recommended that five new RPCs are defined in the MPLS OAM PM model:

o  MPLS-TP Loss Measurement (LM) Message RPC [RFC6375];

o  MPLS-TP Test Message RPC [RFC6375];

o  MPLS-TP Delay Measurement(DM) Message RPC [RFC6375];

## 5.5.  Generic YANG Model extension for NVO3 OAM

### 5.5.1.  Technology Type Extension

No NVO3 technology type has been defined in the LIME base model. Therefore technology type extension is required in the NVO3 OAM model.  The technology type "nvo3" is defined as an identity that augments the base "technology-types" defined in the LIME base model:

```
      identity nvo3{
       base goam:technology-types;
       description
        "nvo3 type";
      }
```

### 5.5.2. Sub Technology Type Extension

In NVO3, since different overlay encapsulation types such as VxLAN, NVGRE can be employed, the "technology-sub-type" data node is defined and added into the NVO3 OAM model to further identify the overlay types within the NVO3 model.  Based on it, we also define a technology sub-type for VxLAN encapsulation.  NVGRE and GENEVE, sub-types can be defined in the same way.

```
   identity technology-sub-type {
             description
               "certain implementations such as nvo3 can have different
                encapsulation types such as vxlan, nvgre and so on.
                Instead of defining separate models for each
                encapsulation, we define a technology sub-type to
                further identify different encapsulations. Technology
                sub-type is associated at the MA level";
           }

        identity technology-sub-type-vxlan {
          base technology-sub-type;
          description
            "technology sub-type is vxlan";
        }
   augment "/goam:domains/goam:domain/goam:MAs/goam:MA" {
          leaf technology-sub-type {
            type identityref {
              base technology-sub-type;
            }
          }
        }
```

### 5.5.3. MEP Configuration Extension

In NVO3, the MEP address is either an IPv4 or IPV6 address.  In the LIME base model, MEP address is defined as an IP address and the same definition can be used for NVO3 with no further modification.

### 5.5.4. Connectivity-Context Extension

In NVO3, one example of context-id is a 24 bit virtual network identifier (VNI). The LIME base model defines a placeholder for context-id. This allows other technologies to easily augment that to include technology specific extensions. The snippet below depicts an example of augmenting context-id to include VNI.

```
augment "/goam:domains/goam:domain/goam:MAs/goam:MA
/goam:connectivity-context"
         {
            case connctivity-context-nvo3 {
              leaf vni {
                type vni;
              }
            }
         }
```

### 5.5.5. RPC Extension

In the NVO3 OAM YANG model, the End-Station-Locator RPC command is defined. This command locates an end-station within the NVO3 deployment. [PTT -- what other tools are applicable??? Presumably one can use ICMP Ping, LSP Ping for CV, and the PM extensions, per RFC 7276 Table 4.]

### 5.5.6. ECMP Extension

In NVO3, flow-entropy depends on the technology sub-type, e.g., VxLAN. Technology sub-type is used to extend the base model to specific usage. The snippet below illustrates the extension for VxLAN.

```
augment "/goam:domains/goam:domain/goam:MAs/goam:MA
/goam:flow-entropy" {
       case flow-entropy-vxlan {
         leaf flags-vxlan {
           type flags-vxlan;
         }
         leaf flow-entropy-vxlan {
           type flow-entropy-vxlan;
         }
       }
     }
```

## 5.6. Generic YANG Model extension for BFD

### 5.6.1. MD Level configuration extension

MD level configuration parameters are management information which can be inherited in the BFD model and set by LIME base model as default values. For example domain name can be set to area-ID in the BFD case. In addition, at the Maintenance Domain level, domain data node at root level can be augmented with technology type and sub-technology type.

Note that MD level configuration parameters provides context information for management system to correlate faults, defects, network failures with location information, which helps quickly identify root causes of network failures. MD level configuration parameters MUST not be carried using BFD protocol since BFD doesn't support transport of these management information.

#### 5.6.1.1. Technology Type Extension

No BFD technology type has been defined in the LIME base model. Therefore a technology type extension is required in the BFD OAM model. The technology type "bfd" is defined as an identity that augments the base "technology-types" defined in the LIME base model:

#### 5.6.1.2. Sub Technology Type Extension

In BFD, since different encapsulation types such as IP/UDP Encapsulation, PW-ACH encapsulation can be employed.

In lime-bfd-extension yang data model, we define an identity: "technology-sub-type" to further identify the encapsulation types within the BFD. And based on it, we also define four identity encapsulation types:

- technology-sub-type-sh-udp: technology sub-type is single hop with IP/UDP encapsulation;

- technology-sub-type-mh-udp: technology sub-type is multiple hop with IP/UDP encasulation;

- technology-sub-type-sh-ach: technology sub-type is single hop with PW-ACH encapsulation;

- technology-sub-type-mh-ach: technology sub-type is multiple hop with PW-ACH encapsulation;

In MD level, we define a sub-technology leaf with an identityref type which base on the technology-sub-type:

```
   augment "/goam:domains/goam:domain/" {
          leaf sub-technology{
            type identityref {
              base technology-sub-type;
            }
          }
        }
```

### 5.6.2. MA configuration extension

MA level configuration parameters are management information which can be inherited in the BFD model and set by LIME base model as default values. In addition, at the Maintenance Association(MA) level, MA data node at the second level can be augmented with connectivity-context extension.

Note that MA level configuration parameters provides context information for management system to correlate faults, defects, network failures with location information, which helps quickly identify root causes of network failures. MA level configuration parameters MUST not be carried using BFD protocol since BFD doesn't support transport of these management information.

#### 5.6.2.1. Connectivity-Context Extension

In BFD, one example of context-id is a 32bit local discriminator. The LIME base model defines a placeholder for context-id. This allows other technologies to easily augment that to include technology specific extensions. The snippet below depicts an example of augmenting context-id to include local discriminator.

```
augment "/goam:domains/goam:domain/goam:MAs/goam:MA
  /goam:connectivity-context"
             {
               case connectivity-context-bfd {
                  leaf local-discriminator {
                    type local-discriminator;
                  }
                }
             }
```

### 5.6.3. MEP configuration extension

In BFD, the MEP address is either an IPv4 or IPV6 address. MEP-ID is either a 2 octet unsigned integer value or a variable length label value. In the LIME base model, MEP-ID is defined as a variable length label value and the same definition can be used for BFD with no further modification. In addition, at the Maintenance Association Endpoint(MEP) level, MEP data node at the third level can be augmented with Session extension and interface extension.

#### 5.6.3.1. Session Configuration Extension

At the Session level, Session data node at the fouth level can be augmented with 3 interval parameters and 2 TTL parameters. In [draft-zheng-bfd-yang], source and destination address in the bfd-session-cfg can be corresponding to Session configuration extension as source MEP and destination MEP.

```
augment /goam:domains/goam:domain/goam:MAs/goam:MA/goam:MEP/goam:session:
+--rw (interval-config-type)?
|  +--:(tx-rx-intervals)
|  |  +--rw desired-min-tx-interval     uint32
|  |  +--rw required-min-rx-interval    uint32
|  +--:(single-interval)
|     +--rw min-interval                uint32

augment /goam:domains/goam:domain/goam:MAs/goam:MA/goam:MEP/goam:session:
+--rw tx-ttl?                     ttl
+--rw rx-ttl                      ttl
```

#### 5.6.3.2. Interface configuration extension

At the Interface level, interface data node at the fifth level can be augmented with the same parameters defined in per-interface configuration of [draft-zheng-bfd-yang].

```
augment /goam:domains/goam:domain/goam:MAs/goam:MA/goam:MEP/goam:session/goam: o
utgoing-interface:
+--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)
|     +--rw min-interval                     uint32
+--rw demand-enabled?                  boolean
+--rw enable-authentication?           boolean
+--rw authentication-parms {bfd-authentication}?
|  +--rw key-chain-name?   string
|  +--rw algorithm?        bfd-auth-algorithm
+--rw desired-min-echo-tx-interval?    uint32
+--rw required-min-echo-rx-interval?   uint32
```

5.6.3.3.  New Notification definition

[GENYANGOAM] defines a notification model which abstracts defects notification in a technology independent manner.However what BFD is required is state change notification, therefore a new notification definition can be specified to meet BFD requirement.

```
notifications:
   +---n state-change-notification
      +--ro local-discriminator?      uint32
      +--ro remote-discriminator?     uint32
      +--ro new-state?                enumeration
      +--ro state-change-reason?      string
      +--ro time-in-previous-state?   string
      +--ro dest-addr?                inet:ip-address
      +--ro source-addr?              inet:ip-address
      +--ro session-cookie?           leafref
      +--ro technology-sub-type?      identityref
      +--ro interface?                leafref
      +--ro echo-enabled?             boolean
```

In this state-change-notification, technology-sub-type is used to identify whether the notification is for single hop or multi-hop or other types.

6.  Open Issues

Do we need to specify usage examples for each technology-specific OAM model?

Applicability of LIME base model structure on BFD in details

Applicability of LIME base model structure on MPLS OAM and MPLS-TP OAM.

## 7. Security Considerations

TBD.

## 8. IANA Considerations

This document registers the following namespace URI in the IETF XML registry.

URI:TBD

## 9. Acknowledgements

The authors would like to thank Gregory Mirsky for his valuable comments and suggestions on this document.

## 10. References

### 10.1. Normative References

[GENYANGOAM]
Senevirathne , T., Finn, N., Kumar, D., Salam, S., Wu, Q., and Z. Wang, "Generic YANG Data Model for Operations, Administration, and Maintenance (OAM)", ID https://datatracker.ietf.org/doc/draft-tissa-lime-yang-oam-model/, June 2015.

[IEEE802.1Q]
"Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks", IEEE Std 802.1Q-2011, August 2011.

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

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

### 10.2. Informative References

[I-D.wang-lime-yang-pm]
Wang, Z., Wu, Q., Kumar, D., and T. Taylor, "Generic YANG Data Model for Operations, Administration, and Maintenance (OAM) Performance Management", draft-wang-lime-yang-pm-01 (work in progress), November 2015.

[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>.

[RFC6291] Andersson, L., van Helvoort, H., Bonica, R., Romascanu, D., and S. Mansfield, "Guidelines for the Use of the "OAM" Acronym in the IETF", BCP 161, RFC 6291, DOI 10.17487/RFC6291, June 2011, <http://www.rfc-editor.org/info/rfc6291>.

[RFC6325] Perlman, R., Eastlake 3rd, D., Dutt, D., Gai, S., and A. Ghanwani, "Routing Bridges (RBridges): Base Protocol Specification", RFC 6325, DOI 10.17487/RFC6325, July 2011, <http://www.rfc-editor.org/info/rfc6325>.

[RFC6371] Busi, I., Ed. and D. Allan, Ed., "Operations, Administration, and Maintenance Framework for MPLS-Based Transport Networks", RFC 6371, DOI 10.17487/RFC6371, September 2011, <http://www.rfc-editor.org/info/rfc6371>.

[RFC7174] Salam, S., Senevirathne, T., Aldrin, S., and D. Eastlake 3rd, "Transparent Interconnection of Lots of Links (TRILL) Operations, Administration, and Maintenance (OAM) Framework", RFC 7174, DOI 10.17487/RFC7174, May 2014, <http://www.rfc-editor.org/info/rfc7174>.

[RFC7276] Mizrahi, T., Sprecher, N., Bellagamba, E., and Y. Weingarten, "An Overview of Operations, Administration, and Maintenance (OAM) Tools", RFC 7276, DOI 10.17487/RFC7276, June 2014, <http://www.rfc-editor.org/info/rfc7276>.

[RFC792] Postel, J., "Internet Control Message Protocol", RFC 792, September 1981.

[TRILLOAMYANG]
Kumar, D., Senevirathne, T., Finn, N., Salam, S., Xia, L., and W. Hao, "YANG Data Model for TRILL Operations, Administration, and Maintenance (OAM) (Work in progress)", May 2015.

## Appendix A. Contributing Authors Infomation

Huub van Helvoort
Hai Gaoming BV
Netherlands

Email: huubatwork@gmail.com

Roland Schott
Deutsche Telekom
Deutsche-Telekom-Allee 7
Darmstadt 64295
Germany

EMail: Roland.Schott@telekom.de

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Deepak Kumar
CISCO Systems
510 McCarthy Blvd
Milpitas, CA 95035
USA

Email: dekumar@cisco.com

Yuji Tochio
Fujitsu
Japan

Email: tochio@jp.fujitsu.com

Guangying Zheng
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: zhengguangying@huawei.com

Daniel King
Lancaster University

UK

Email: daniel@olddog.co.uk

Zitao Wang
Huawei Technologies,Co.,Ltd
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: wangzitao@huawei.com

Authors' Addresses

Tom Taylor (editor)
PT Taylor Consulting
Ottawa
Canada

Email: tom.taylor.stds@gmail.com

Yan Zhuang (editor)
Huawei Technologies,Co.,Ltd
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: zhuangyan.zhuang@huawei.com