Abstract

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

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 20, 2016.
Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction .................................................. 3
   1.1. Contributors ................................... 3
2. Design of the Data Model ...................................... 3
   2.1. Design of Configuration Model ......................... 4
       2.1.1. Single-hop IP .............................. 4
       2.1.2. Multi-hop IP ............................. 5
       2.1.3. Traffic Engineering Tunnels ............... 6
       2.1.4. LDP Label Switched Paths ................. 6
       2.1.5. Link Aggregation Groups .................. 6
       2.1.6. Per-interface Configuration ............... 7
   2.2. Design of Operational Model ............................ 7
   2.3. Notifications ........................................ 8
   2.4. RPC Operations ..................................... 8
   2.5. BFD Configuration Data Hierarchy ..................... 8
       2.5.1. Centralized BFD Configuration ............ 8
       2.5.2. Configuration in BFD clients ............. 9
   2.6. Operational Data Hierarchy ............................ 11
   2.7. Notifications ....................................... 14
   2.8. Examples ............................................ 16
   2.9. Interaction with other YANG modules .................. 16
   2.10. BFD Yang Module .................................... 16
   2.11. BFD Client Example Configuration Yang Module .... 31
   2.12. Security Considerations ............................. 33
   2.13. IANA Considerations ................................. 33
   2.14. Acknowledgements ................................... 33
3. References .................................................... 33
   3.1. Normative References .................................. 33
   3.2. Informative References ............................... 34
Appendix A. Change log .......................................... 35
   A.1. Changes between versions -03 and -04 ............... 35
   A.2. Changes between versions -02 and -03 ............... 35
1. Introduction

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

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

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

2) Two systems connected via multiple hops [RFC5883]

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

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

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

1.1. Contributors

2. Design of the Data Model

The BFD YANG data model follows a VRF-centric model by augmenting the "routing-protocol" data node in [I-D.ietf-netmod-routing-cfg].

2.1. Design of Configuration Model

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

Some implementations have BFD session configuration under the BFD clients. For example, BFD session configuration is 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. So the configuration model has groupings, containing multiplier and interval(s), which are to be used by BFD clients. Other parameters which remain under BFD control, e.g. demand mode and authentication, are configured under BFD.

We provide groupings, which contain the basic BFD session parameters, for applications to use. This ensures consistency of BFD parameters across applications.

Although [RFC5880] allows for different values for transmit and receive intervals, many 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. Our 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".

2.1.1. Single-hop IP

For single-hop IP, the BFD multiplier and interval(s) are configured in BFD clients. It is highly desirable to have the BFD configuration consistent between those clients. Therefore, we have a grouping for BFD configuration that applications can import in their YANG module:

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

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

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

The configuration items are:

enabled
Set to true to enable BFD.

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

min-interval
This is the Desired Min TX Interval and Required Min RX Interval as defined in [RFC5880].

OR

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

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

2.1.2. Multi-hop IP

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

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

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

Since we are following a VRF-centric model we do not need a VRF field in the key.

Here is the list of parameters:

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

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

required-min-rx-interval

This is the Required Min RX Interval as defined in [RFC5880].

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

enable-authentication
Set to true to enable BFD authentication.

authentication-algorithm
Authentication algorithm to use, if enabled.

key-chain-name
Key-chain to be used for authentication, if enabled.

tx-ttl
TTL of outgoing BFD control packets.

rx-ttl
Minimum TTL of incoming BFD control packets.

2.1.3. Traffic Engineering Tunnels

For Traffic Engineering (TE) tunnels, BFD is configured under the TE tunnel since it is the Traffic Engineering application which knows the desired failure detection parameters. The grouping for BFD clients in Section 2.1.1 can be used by the TE application for the basic BFD parameters. 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 TE application.

2.1.4. LDP Label Switched Paths

For LDP LSPs, BFD is configured under the LDP FEC. The grouping for BFD clients in Section 2.1.1 can be used by the LDP application for the basic BFD parameters. For BFD parameters which are specific to the LDP application, if any, these parameters will be defined in the YANG model of the LDP application.

2.1.5. Link Aggregation Groups

Per [RFC7130], configuring BFD on LAG consists of having micro-BFD sessions on each LAG member link. The grouping for BFD clients in Section 2.1.1 can be used by the LAG application for the basic BFD parameters. This grouping can be used per-LAG or per member-link. For BFD parameters which are specific to the LAG application, e.g. the IP address of the peer system which can be reached by the LAG,
the parameters will be defined in the YANG model of the LAG application.

2.1.6. Per-interface Configuration

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

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

- enable-authentication
  Set to true to enable BFD authentication.

- authentication-algorithm
  Authentication algorithm to use, if enabled.

- key-chain-name
  Key-chain to be used for authentication, if enabled.

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

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

2.2. Design of Operational Model

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

For some session types, there may be more than 1 session on the virtual path to the destination. For example, with IP multi-hop there could be multiple BFD sessions from the source to the same destination to test the various paths (ECMP) to the destination. Each of the BFD sessions on the same virtual path is uniquely identified by the source UDP port.

2.3. Notifications

This YANG model defines a list of notifications to inform clients of BFD 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, routing-instance and the reason that the BFD session state changed.

2.4. RPC Operations

TBD

2.5. BFD Configuration Data Hierarchy

2.5.1. Centralized BFD Configuration

The following is the centralized configuration data hierarchy:

We have a container which contains a list for each session type. This contains session configuration for only IP multi-hop sessions.

We have per-interface configuration for authentication, demand-mode, etc. This is used e.g. for IP single-hop sessions whose base BFD configuration belongs to the BFD clients.
module: ietf-bfd
augment /rt:routing/rt:routing-instance/rt:routing-protocols/
 rt:routing-protocol:
 ++--rw bfd
  ++--rw bfd-cfg
   ++--rw bfd-session-cfg
    ++--rw session-ip-mh* [source-addr dest-addr]
     ++--rw source-addr inet:ip-address
     ++--rw dest-addr inet:ip-address
     ++--rw admin-down? 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)
        ++--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 tx-ttl? ttl
     ++--rw rx-ttl ttl
  ++--rw bfd-interface-cfg* [interface] {bfd-interface-config}?
   ++--rw interface if:interface-ref
   ++--rw local-multiplier? multiplier
   ++--rw (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

2.5.2. Configuration in BFD clients

The following is the configuration data hierarchy for a hypothetical BFD client called example-bfd-routing-app. The BFD configuration is supported conditionally via use of if-feature.
We have a list of areas and in each area we have a list of interfaces. The BFD configuration grouping is used in a hierarchical fashion, it can be applied in "area" and "interface":

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

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

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

- The tx and rx intervals can be configured separately or a single interval can be configured.

module: example-bfd-routing-app
  +--rw area* [area-id]
    +--rw area-id uint32
    +--rw bfd-cfg
      +--rw enabled? 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)
          | +--rw min-interval uint32
    +--rw interface* [interface]
      +--rw interface if:interface-ref
        +--rw bfd-cfg
          +--rw enabled? 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)
              | +--rw min-interval uint32
2.6. Operational Data Hierarchy

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

module: ietf-bfd
augment /rt:routing/rt:routing-instance/rt:routing-protocols/rt:routing-protocol:
  +-rw bfd
     +-ro bfd-oper
        +-ro bfd-session-statistics
          +-ro ip-sh-session-num? uint32
          +-ro ip-mh-session-num? uint32
          +-ro total-session-num? uint32
          +-ro session-up-num? uint32
          +-ro sess-down-num? uint32
          +-ro sess-admin-down-num? uint32
        +-ro bfd-session-lists
          +-ro session-ip-sh* [interface dest-addr]
            +-ro interface if:interface-ref
            +-ro dest-addr inet:ip-address
            +-ro source-addr? inet:ip-address
            +-ro session-type? bfd-session-type
            +-ro local-discriminator? discriminator
            +-ro remote-discriminator? discriminator
            +-ro remote-multiplier? multiplier
            +-ro out-interface? if:interface-ref
            +-ro demand-capability? boolean
            +-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? diagnostic
            +-ro remote-diagnostic? diagnostic
            +-ro detection-mode? enumeration
            +-ro negotiated-tx-interval? uint32
            +-ro negotiated-rx-interval? uint32
            +-ro negotiated-echo-tx-interval? uint32
            +-ro detection-time? uint32
          +-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? uint32
            +-ro admin-down-count? uint32
```yang
++--ro receive-packet-count? uint64
++--ro send-packet-count? uint64
++--ro receive-bad-packet? uint64
++--ro send-failed-packet? uint64

++--ro session-ip-mh-group* [source-addr dest-addr]
  ++--ro source-addr inet:ip-address
  ++--ro dest-addr inet:ip-address

++--ro session-ip-mh* [source-port]
  ++--ro ttl? ttl
  ++--ro session-type? bfd-session-type
  ++--ro local-discriminator? discriminator
  ++--ro remote-discriminator? discriminator
  ++--ro remote-multiplier? multiplier
  ++--ro out-interface? if:interface-ref
  ++--ro demand-capability? boolean
  ++--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? diagnostic
  ++--ro remote-diagnostic? diagnostic
  ++--ro detection-mode? enumeration
  ++--ro negotiated-tx-interval? uint32
  ++--ro negotiated-rx-interval? uint32
  ++--ro negotiated-echo-tx-interval? uint32
  ++--ro detection-time? uint32

++--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? uint32
  ++--ro admin-down-count? uint32
  ++--ro receive-packet-count? uint64
  ++--ro send-packet-count? uint64
  ++--ro receive-bad-packet? uint64
  ++--ro send-failed-packet? uint64

++--ro session-te-tunnel* [tunnel-name]
  ++--ro tunnel-name string
  ++--ro session-type? bfd-session-type
  ++--ro local-discriminator? discriminator
  ++--ro remote-discriminator? discriminator
  ++--ro remote-multiplier? multiplier
  ++--ro out-interface? if:interface-ref
  ++--ro demand-capability? boolean
  ++--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? diagnostic |
|   +--ro remote-diagnostic? diagnostic |
|   +--ro detection-mode? enumeration |
|   +--ro negotiated-tx-interval? uint32 |
|   +--ro negotiated-rx-interval? uint32 |
|   +--ro negotiated-echo-tx-interval? uint32 |
|   +--ro detection-time?    uint32 |
| +--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?       uint32 |
|   +--ro admin-down-count? uint32 |
|   +--ro receive-packet-count? uint64 |
|   +--ro send-packet-count? uint64 |
|   +--ro receive-bad-packet? uint64 |
|   +--ro send-failed-packet? uint64 |
| +--ro ldp-fec            inet:ip-prefix |
| +--ro session-lsp-group* [ldp-fec] |
|   +--ro ttl?             ttl |
|   +--ro session-type? bfd-session-type |
|   +--ro local-discriminator? discriminator |
|   +--ro remote-discriminator? discriminator |
|   +--ro remote-multiplier? multiplier |
|   +--ro out-interface? if:interface-ref |
|   +--ro demand-capability? boolean |
|   +--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? diagnostic |
|   +--ro remote-diagnostic? diagnostic |
|   +--ro detection-mode? enumeration |
|   +--ro negotiated-tx-interval? uint32 |
|   +--ro negotiated-rx-interval? uint32 |
|   +--ro negotiated-echo-tx-interval? uint32 |
|   +--ro detection-time?    uint32 |
| +--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 session-lag* [lag-name]
  +--ro lag-name                    if:interface-ref
  +--ro session-lag-micro* [member-link]
    +--ro member-link                if:interface-ref
    +--ro session-type?              bfd-session-type
    +--ro local-discriminator?       discriminator
    +--ro remote-discriminator?       discriminator
    +--ro remote-multiplier?          multiplier
    +--ro out-interface?              if:interface-ref
    +--ro demand-capability?          boolean
    +--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?           diagnostic
    +--ro remote-diagnostic?          diagnostic
    +--ro detection-mode?             enumeration
    +--ro negotiated-tx-interval?     uint32
    +--ro negotiated-rx-interval?     uint32
    +--ro negotiated-echo-tx-interval? uint32
    +--ro detection-time?             uint32
  +--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?                uint32
    +--ro admin-down-count?          uint32
    +--ro receive-packet-count?      uint64
    +--ro send-packet-count?         uint64
    +--ro receive-bad-packet?        uint64
    +--ro send-failed-packet?        uint64

2.7. Notifications

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

module: ietf-bfd
augment /rt:routing/rt:routing-instance/rt:routing-protocols/

rt:routing-protocol:
notifications:
  +++-n bfd-singlehop-notification
   |  +--ro local-discr?  discriminator
   |  +--ro remote-discr? discriminator
   |  +--ro new-state?    state
   |  +--ro state-change-reason? string
   |  +--ro time-in-previous-state? string
   |  +--ro dest-addr?    inet:ip-address
   |  +--ro source-addr?  inet:ip-address
   |  +--ro session-index? uint32
   |  +--ro session-type? bfd-session-type
   |  +--ro interface?    if:interface-ref
   |  +--ro echo-enabled? boolean
  +++-n bfd-multihop-notification
   |  +--ro local-discr?  discriminator
   |  +--ro remote-discr? discriminator
   |  +--ro new-state?    state
   |  +--ro state-change-reason? string
   |  +--ro time-in-previous-state? string
   |  +--ro dest-addr?    inet:ip-address
   |  +--ro source-addr?  inet:ip-address
   |  +--ro session-index? uint32
   |  +--ro session-type? bfd-session-type
  +++-n bfd-te-tunnel-notification
   |  +--ro local-discr?  discriminator
   |  +--ro remote-discr? discriminator
   |  +--ro new-state?    state
   |  +--ro state-change-reason? string
   |  +--ro time-in-previous-state? string
   |  +--ro dest-addr?    inet:ip-address
   |  +--ro source-addr?  inet:ip-address
   |  +--ro session-index? uint32
   |  +--ro session-type? bfd-session-type
   |  +--ro tunnel-name?  string
  +++-n bfd-ldp-lsp-notification
   |  +--ro local-discr?  discriminator
   |  +--ro remote-discr? discriminator
   |  +--ro new-state?    state
   |  +--ro state-change-reason? string
   |  +--ro time-in-previous-state? string
   |  +--ro dest-addr?    inet:ip-address
   |  +--ro source-addr?  inet:ip-address
   |  +--ro session-index? uint32
   |  +--ro session-type? bfd-session-type
   |  +--ro ldp-fec?      inet:ip-prefix
   |  +--ro source-port?  inet:port-number
  +++-n bfd-lag-notification
2.8. Examples

2.9. Interaction with other YANG modules

TBD.

2.10. BFD Yang Module

<CODE BEGINS> file "ietf-bfd@2015-07-01.yang"
module ietf-bfd {
    namespace "urn:ietf:params:xml:ns:yang:ietf-bfd";
    // replace with IANA namespace when assigned
    prefix "bfd";

    import ietf-interfaces {
        prefix "if";
    }
    import ietf-inet-types {
        prefix "inet";
    }
    import ietf-yang-types {
        prefix "yang";
    }
    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>
        WG Chair: Jeff Haas
        WG Chair: Nobo Akiya
        Editor:   Lianshu Zheng and Reshad Rahman";
    description
        "This module contains the YANG definition for BFD parameters as
         per RFC5880, RFC5881 and RFC5883";
    revision 2015-07-01 {
        description "Initial revision."
        reference "RFC XXXX: A YANG data model for BFD";
identity bfd {
  base "rt:routing-protocol";
  description "BFD protocol";
}

typedef discriminator {
  type uint32 {
    range 1..4294967295;
  }
  description "BFD discriminator";
}

typedef diagnostic {
  type enumeration {
    enum none {
      value 0;
      description "None";
    }
    enum controlExpiry {
      value 1;
      description "Control timer expiry";
    }
    enum echoFailed {
      value 2;
      description "Echo failure";
    }
    enum nborDown {
      value 3;
      description "Neighbor down";
    }
    enum fwdingReset {
      value 4;
      description "Forwarding reset";
    }
    enum pathDown {
      value 5;
      description "Path down";
    }
    enum concPathDown {
      value 6;
      description "Concatenated path down";
    }
    enum adminDown {
      value 7;
      description "Admin down";
    }
    enum reverseConcPathDown {
      value 8;
      description "Reverse concatenated path down";
    }
  }
}
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";
}

typedef multiplier {
    type uint8 {
        range 1..255;
    }
    description "Multiplier";
}

typedef ttl {
    type uint8 {
        range 1..255;
    }
    description "Time To Live";
}

typedef bfd-session-type {
    type enumeration {
        enum ip-single-hop {
            description "IP single hop";
        }
        enum ip-multi-hop {
            description "IP multi hop";
        }
        enum te-tunnel {
            description "Traffic Engineering tunnles";
        }
    }
}
enum ldp-lsp {
   description "LDP Label Switched Path";
}
enum lag {
   description "Micro-BFD on LAG member links";
}

description
   "BFD session type, this indicates the path type that BFD is
   running on";

typedef bfd-auth-algorithm {
   type enumeration {
      enum simple-password {
         description
            "Simple password";
      }
      enum keyed-md5 {
         description
            "Keyed message Digest 5";
      }
      enum meticulous-keyed-md5 {
         description
            "Meticulous keyed message Digest 5";
      }
      enum keyed-sha-1 {
         description
            "Keyed secure hash algorithm (SHA1) ";
      }
      enum meticulous-keyed-sha-1 {
         description
            "Meticulous keyed secure hash algorithm (SHA1) ";
      }
   }
   description "Authentication algorithm";
}
leaf local-multiplier {
  type multiplier;
  default 3;
  description "Multiplier transmitted by local system";
}

choice interval-config-type {
  description "Two interval values or 1 value used for both tx and rx";
  case tx-rx-intervals {
    leaf desired-min-tx-interval {
      type uint32;
      units microseconds;
      mandatory true;
      description "Desired minimum transmit interval of control packets";
    }
    leaf required-min-rx-interval {
      type uint32;
      units microseconds;
      mandatory true;
      description "Required minimum receive interval of control packets";
    }
  }
  case single-interval {
    leaf min-interval {
      type uint32;
      units microseconds;
      mandatory true;
      description "Desired minimum transmit interval and required " +
      "minimum receive interval of control packets";
    }
  }
}

grouping bfd-grouping-common-cfg-parms {
  description "BFD grouping for common config parameters";
  uses bfd-grouping-base-cfg-parms;
  leaf demand-enabled {
    type boolean;
    default false;
    description "To enable demand mode";
  }
  leaf enable-authentication {
    type boolean;
    default false;
  }
}
description
"If set, the Authentication Section is present and the
session is to be authenticated (see RFC5880 section 6.7
for details).";
}
corner authentication-parms {
  if-feature bfd-authentication;
  description "Parameters for authentication";
  leaf key-chain-name {
    type string;
    must ".../algorithm" {
      error-message
        "May not be configured without algorithm";
      description "Requires algorithm";
    }
    description
      "Key chain name";
  }
  leaf algorithm {
    type bfd-auth-algorithm;
    must ".../key-chain" {
      error-message
        "May not be configured without key-chain";
      description "Requires key-chain";
    }
    description "Authentication algorithm to be used";
  }
}

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

grouping bfd-client-base-cfg-parms {
  description
    "BFD grouping for base config parameters which could be used
    by a protocol which is a client of BFD";
}
container bfd-cfg {
    description "BFD configuration";
    leaf enabled {
        type boolean;
        default false;
        description "True if BFD is enabled";
    }
    uses bfd-grouping-base-cfg-parms;
}

grouping bfd-all-session {
    description "BFD session operational information";
    leaf session-type {
        type bfd-session-type;
        description "BFD session type, this indicates the path type that BFD is running on";
    }
    leaf local-discriminator {
        type discriminator;
        description "Local discriminator";
    }
    leaf remote-discriminator {
        type discriminator;
        description "Remote discriminator";
    }
    leaf remote-multiplier {
        type multiplier;
        description "Remote multiplier";
    }
    leaf out-interface {
        type if:interface-ref;
        description "Outgoing physical interface name";
    }
    leaf demand-capability {
        type boolean;
        description "Local demand mode capability";
    }
    leaf source-port {
        type inet:port-number;
        description "Source UDP port";
    }
    leaf dest-port {
        type inet:port-number;
        description "Destination UDP port";
    }
    list session-running {
        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 diagnostic;
    description "Local diagnostic";
}
leaf remote-diagnostic {
    type diagnostic;
    description "Remote diagnostic";
}
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 negotiated-echo-tx-interval {
  type uint32;
  units microseconds;
  description "Negotiated echo transmit interval";
}
leaf detection-time {
  type uint32;
  units microseconds;
  description "Detection time";
}
} 
list session-statistics {
  description "BFD session statistics";
  leaf create-time {
    type yang:date-and-time;
    description
      "Time and date when session was created";
  }
  leaf last-down-time {
    type yang:date-and-time;
    description
      "Time and date of last time the session went down";
  }
  leaf last-up-time {
    type yang:date-and-time;
    description
      "Time and date of last time the session went up";
  }
  leaf down-count {
    type uint32;
    description "Session Down Count";
  }
  leaf admin-down-count {
    type uint32;
    description "Session Admin-Down Count";
  }
  leaf receive-packet-count {
    type uint64;
    description "Received Packet Count";
  }
  leaf send-packet-count {
    type uint64;
    description "Sent Packet Count";
  }
leaf receive-bad-packet {
  type uint64;
  description "Received bad packet count";
}
leaf send-failed-packet {
  type uint64;
  description "Packet Failed to Send Count";
}

augment "/rt:routing/rt:routing-instance/rt:routing-protocols/"
  + "rt:routing-protocol" {
    when "rt:type = 'bfd:bfd'" {
      description
        "This augment is only valid for a protocol instance
         of BFD.";
    }
    description "BFD augmentation.";
    container bfd {
      description "BFD top-level container";
      container bfd-cfg {
        description "BFD configuration";
        container bfd-session-cfg {
          description "BFD session configuration";
          list session-ip-mh {
            key "source-addr dest-addr";
            description "List of IP multi-hop sessions";
            leaf source-addr {
              type inet:ip-address;
              description "Local IP address";
            }
            leaf dest-addr {
              type inet:ip-address;
              description "IP address of the peer";
            }
            leaf admin-down {
              type boolean;
              default false;
              description "Is the BFD session administratively down";
            }
          }
        }
      }
    }
  }
  uses bfd-grouping-common-cfg-parms;
leaf tx-ttl {
  type ttl;
  default 255;
  description "TTL of outgoing BFD control packets";
leaf rx-ttl {
  type ttl;
  mandatory true;
  description "Minimum allowed TTL value for incoming BFD control packets";
}
}
list bfd-interface-cfg {
  if-feature bfd-interface-config;
  key interface;
  description "Per-interface BFD configuration";
  leaf interface {
    type if:interface-ref;
    description "Interface";
  }
  uses bfd-grouping-common-cfg-parms;
  uses bfd-grouping-echo-cfg-parms;
}
container bfd-oper {
  config "false";
  description "BFD operational container";
  container bfd-session-statistics {
    description "BFD session counters";
    leaf ip-sh-session-num {
      type uint32;
      description "IP single hop session number";
    }
    leaf ip-mh-session-num {
      type uint32;
      description "IP multi hop session Number";
    }
    leaf total-session-num {
      type uint32;
      description "Total session number";
    }
    leaf session-up-num {
      type uint32;
      description "Session up number";
    }
    leaf sess-down-num {
      type uint32;
      description "Session down number";
    }
    leaf sess-admin-down-num {
type uint32;
description "Session admin-down number";
}
)

container bfd-session-lists {

description
"Contains multiple session lists, one per type";
list session-ip-sh {
  key "interface dest-addr";
  description "BFD 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 "BFD peer address";
  }
  leaf source-addr {
    type inet:ip-address;
    description "BFD source address";
  }
  uses bfd-all-session;
}
list session-ip-mh-group {
  key "source-addr dest-addr";
  description
  "BFD IP multi-hop group of sessions. A group of " +
  "sessions is between 1 source and 1 destination, " +
  "each session uses a different source UDP port for " +
  "ECMP.";
  leaf source-addr {
    type inet:ip-address;
    description "BFD source address";
  }
  leaf dest-addr {
    type inet:ip-address;
    description "BFD peer address";
  }
list session-ip-mh {
  key "source-port";
  description
  "The BFD sessions between a source and a. " +
  "destination. Source UDP port is unique for " +
  "each session in the group.";
  leaf ttl {

type ttl;
  description "TTL of outgoing packets";
} uses bfd-all-session;
}

list session-te-tunnel {
  key "tunnel-name";
  description "BFD over TE tunnel";
  leaf tunnel-name {
    type string;
    description "Name of TE tunnel";
  }
  uses bfd-all-session;
}

list session-ldp-lsp-group {
  key "ldp-fec";
  description "BFD over LDP LSP group of sessions. A group of " + 
    "sessions is to one LDP FEC, each session uses a " + 
    "different source UDP port for ECMP.";
  leaf ldp-fec {
    type inet:ip-prefix;
    description "LDP FEC";
  }
}

list session-ldp-lsp {
  key "source-port";
  description "The BFD sessions on an LDP FEC. Source UDP " + 
    "port is unique for each session in the group.";
  leaf ttl {
    type ttl;
    description "TTL of outgoing packets";
  }
  uses bfd-all-session;
}

list session-lag {
  key "lag-name";
  description "A LAG interface on which BFD is running";
  leaf lag-name {
    type if:interface-ref;
    description "Name of the LAG";
  }
  list session-lag-micro {
    key "member-link";
    description "Micro-BFD over LAG. This represents BFD " +
  
  }
"over one member link";
leaf member-link {
type if:interface-ref;
description
"Member link on which micro-BFD is running";
}
uses bfd-all-session;
}
}
}
}
}
}

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

notification bfd-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-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";
    }
}

notification bfd-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-notification-parms;
}
notification bfd-te-tunnel-notification {
    description
        "Notification for BFD over TE tunnel session state change. " +
        "An implementation may rate-limit notifications, e.g. when a" +
        "session is continuously changing state.";
    uses bfd-notification-parms;

leaf tunnel-name {
  type string;
  description "TE tunnel to which this BFD session belongs to";
}

notification bfd-ldp-lsp-notification {
  description
    "Notification for BFD over LDP LSP session state change. " +
    "An implementation may rate-limit notifications, e.g. when a" +
    "session is continuously changing state.";
  uses bfd-notification-parms;

  leaf ldp-fec {
    type inet:ip-prefix;
    description "LDP FEC";
  }
  leaf source-port {
    type inet:port-number;
    description "Source UDP port";
  }
}

notification bfd-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-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";
  }
}

2.11. BFD Client Example Configuration Yang Module

module example-bfd-routing-app {
  namespace "urn:ietf:params:xml:ns:yang:example-bfd-routing-app";
  prefix bfd-routing-app;

  import ietf-bfd {
    prefix "bfd";
  }

import ietf-interfaces {
    prefix "if";
}

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

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

revision 2015-07-01 {
    description "Initial revision.";
    reference "RFC XXXX: An example BFD routing application";
}

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

list area {
    key "area-id";
    description "Specify a routing area.";

    leaf area-id {
        type uint32;
        description "Area";
    }

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

    list interface {
        key "interface";
        description "List of interfaces";
        leaf interface {
            type if:interface-ref;
            description "Interface";
        }

        uses bfd:bfd-client-base-cfg-parms {
            if-feature routing-app-bfd;
        }
    }
}
2.12. Security Considerations

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

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

2.13. IANA Considerations

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

URI: TBD

2.14. Acknowledgements

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

3. References

3.1. Normative References

[ RFC2119 ] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,


3.2. Informative References

[I-D.ietf-netconf-restconf]

[I-D.ietf-netmod-routing-cfg]
Appendix A. Change log

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

A.1. Changes between versions -03 and -04
   o Follow VRF-centric model
   o IP single-hop session configuration in BFD clients

A.2. Changes between versions -02 and -03
   o Fixed date mismatch
   o Updated authors

A.3. Changes between versions -01 and -02
   o Fixed errors and warnings from "pyang --ietf"
   o Added appendix for "Change log"

A.4. Changes between versions -00 and -01

   In the YANG module section:
   o Added missing filename
   o Added missing CODE ENDS

Authors' Addresses

Lianshu Zheng (editor)
Huawei Technologies
China

Email: vero.zheng@huawei.com

Reshad Rahman (editor)
Cisco Systems
Canada

Email: rrahman@cisco.com
Layer-Transcending Traceroute for Overlay Networks like VXLAN
draft-nordmark-nvo3-transcending-traceroute-01

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 is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 3, 2016.
Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the
document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal
Provisions Relating to IETF Documents
(http://trustee.ietf.org/license-info) in effect on the date of
publication of this document. Please review these documents
carefully, as they describe your rights and restrictions with respect
to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction .................................................. 3
2. Solution Overview ............................................. 4
3. Goals and Requirements ....................................... 5
4. Definition Of Terms ............................................ 6
5. Example Topologies ............................................. 6
6. Controlling and selecting ttl behavior ....................... 10
7. Introducing a ttl copyin flag in the encapsulation header .... 10
8. Encapsulation Behavior ........................................ 11
9. Decapsulating Behavior ........................................ 14
10. Other ICMP errors .............................................. 15
11. Security Considerations ..................................... 15
12. IANA Considerations .......................................... 16
13. Acknowledgements ............................................. 16
14. References .................................................... 16
    14.1. Normative References .................................... 16
    14.2. Informative References .................................. 16
Authors’ Addresses .............................................. 18
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:

- 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.

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 straightforward.

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]. Such extensions are for further study.

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]:

- **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.

- **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.

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

- **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.

- **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.
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
2  2.0.1.1 (2.0.1.1)  1.104 ms  1.235 ms  1.729 ms
3  2.0.1.2 (2.0.1.2)  2.106 ms  2.007 ms  2.156 ms
4  2.0.2.1 (2.0.2.1)  35.034 ms  24.490 ms  21.626 ms
5  1.0.1.2 (1.0.1.2)  40.830 ms  44.694 ms  75.620 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 |  | 1.0.3.4 |
|         |                | 1.0.1.2 |  | 1.0.3.3 |  |         |
```

---

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
```
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.,

```plaintext
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.,

```plaintext
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.

```
+---------------------------------+---------------------------------+
| R | R | R | R | I | 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:

- 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.

- 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.

- 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.
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.
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:

- 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.

- 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.

- 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. 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 presents 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.
12. IANA Considerations

TBD

13. Acknowledgements

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

14. References

14.1. Normative References


14.2. Informative References

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


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.


Bonica, R., Gan, D., Tappan, D., and C. Pignataro,


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
Applicability of Generic YANG Data Model for layer Independent OAM Management
draft-zhuang-lime-yang-oam-model-applicability-01

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

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 10, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents
carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ........................................... 3
2. Conventions Used In This Document ..................... 3
   2.1. Terminology ...................................... 3
3. Basic Structure of Generic YANG Model for OAM ........ 4
   3.1. Performance Management Support ...................... 6
4. Guidelines For Extending the LIME Base Data Model .... 6
   4.1. Extend configuration structure with technology specific parameters ......................... 7
      4.1.1. Maintenance domain (MD) at the root level ........ 8
      4.1.2. Maintenance Association (MA) at the second level .. 9
      4.1.3. Maintenance Association Endpoint (MEP) at the third level ..................... 10
      4.1.4. Session at the fourth level .................... 10
      4.1.5. Interface at the fifth level ................... 10
   4.2. Extend RPC structure with technology specific parameters ................................ 10
   4.3. Extend Notification structure with technology specific parameters ......................... 12
   4.4. Define New RPCs and Notifications ................... 12
5. Applicability of LIME Model to Various Technologies ... 12
   5.1. Generic YANG Model extension for IP OAM .......... 13
      5.1.1. MD Configuration Extension ....................... 13
      5.1.2. MA Configuration Extension ....................... 13
      5.1.3. MEP Configuration Extension ...................... 14
      5.1.4. RPC Extension ................................ 14
      5.1.5. Performance Monitoring Extension ................. 15
   5.2. Generic YANG Model extension for TRILL OAM ....... 16
      5.2.1. MD Configuration Extension ....................... 16
      5.2.2. MA Configuration Extension ....................... 16
      5.2.3. MEP Configuration Extension ...................... 17
      5.2.4. RPC Extension ................................ 18
      5.2.5. Performance Management (PM) Extension .......... 19
      5.2.6. Usage example ................................ 19
   5.3. Generic YANG Model extension for MPLS OAM ....... 23
      5.3.1. MD Configuration Extension ....................... 23
      5.3.2. MA Configuration Extension ....................... 24
      5.3.3. MEP Configuration Extension ...................... 25
      5.3.4. RPC Extension ................................ 26
      5.3.5. Performance Management Extension ................. 26
      5.3.6. Usage Example ................................ 27
   5.4. Generic YANG Model extension for MPLS-TP OAM ... 28
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].
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.
The generic YANG OAM model comprises three definitions for configuration and operational state data:

- configuration model definition;
- Remote procedure call (RPC) definition;
- 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:

```yml
/* MEP Configuration extension */
augment /goam:domains/goam:domain/goam:MAs/goam:MA/goam:MEP:
  +--rw delay-measurements?
  +--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 augment 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:

```yancy
module 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:

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

```Yang
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:

- **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";
  }
  ```

- **Output data node**

  Similarly, output data node 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
/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:

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

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

```
  /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.
    /goam:flow-entropy:
        +--:(flow-entropy-trill)
            +--rw flow-entropy-trill?  flow-entropy-trill
    /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:

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

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

![TRILL OAM scenario](https://example.com/image)

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
  /goam:session/goam:connectivity-context:
    +++:(connectivity-context-vlan)
    | | +--rw connectivity-context-vlan? vlan
    +++:(connectivity-context-fgl)
```
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 </technology>
<MD-name-string> MD1 </MD-name-string>
<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>
</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>
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:

```yaml
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:MAas/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.

```yml
augment "/goam:domains/goam:domain/goam:MA\s/goam:MA\s/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];
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.

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:

```yaml
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:MAas/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:MAsgoam: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:
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.

```yson
augment "/goam:domains/goam:domain/goam:MAs/goam:MA
/goam:connectivity-context"
{
    case connectivity-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.

```yson
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 encapsulation;
- "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:

```xml
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.

```xml
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 fourth 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].
+--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]

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


10.2. Informative References


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 Information

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