Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 20, 2016

L. Zheng, Ed.
Huawei Technologies
R. Rahman, Ed.
Cisco Systems
S. Pallagatti
Juniper Networks
M. Jethanandani
Cisco Systems
G. Mirsky
Ericsson

August 19, 2015

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

Abstract

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

Requirements Language

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

Status of This Memo

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

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

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

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

Table of Contents

1. Introduction

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

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

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

2) Two systems connected via multiple hops [RFC5883]

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

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

BFD typically does not operate on its own. Various control protocols, 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 suport 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 Enginnering 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 sesssion-statistics*
               |     +--ro create-time?           yang:date-and-time
               |     +--ro last-down-time?        yang:date-and-time
               |     +--ro last-up-time?          yang:date-and-time
               |     +--ro 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-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 sesssion-statistics*
|        +--ro create-time?            yang:date-and-time
|        +--ro last-down-time?         yang:date-and-time
|        +--ro last-up-time?           yang:date-and-time
|        +--ro 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 sesssion-statistics*
|     +--ro create-time?            yang:date-and-time
|     +--ro last-down-time?         yang:date-and-time
|     +--ro last-up-time?           yang:date-and-time
|     +--ro 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-ldp-lsp-group* [ldp-fec]
|  +--ro ldp-fec            inet:ip-prefix
|  +--ro session-ldp-lsp* [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 sesssion-statistics*
|        +--ro create-time?            yang:date-and-time
|        +--ro last-down-time?         yang:date-and-time
|        +--ro last-up-time?           yang:date-and-time
```

```
|        +--ro 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-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 sesssion-statistics*
         +--ro create-time?            yang:date-and-time
         +--ro last-down-time?         yang:date-and-time
         +--ro last-up-time?           yang:date-and-time
         +--ro 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
```

```
         +--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
```

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";
```

```
        }
      }
      description "BFD diagnostic";
    }
    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 tunnes";
        }
```

```
          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";
      }
      feature bfd-interface-config {
        description "BFD per-interface config supported";
      }
      feature bfd-authentication {
        description "BFD authentication supported";
      }
      grouping bfd-grouping-base-cfg-parms {
        description "BFD grouping for base config parameters";
```

```
      leaf local-multiplier {
        type multiplier;
        default 3;
        description "Multiplier transmitted by local system";
      }
      choice interval-config-type {
        description
          "Two interval values or 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 enbale 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).";
      }
      container 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 minumum 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 sesssion-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";
        }
      }
    }
    <CODE ENDS>
```

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 as assign a new new namespace URI from the IETF XML registry.

URI:TBD

2.14. Acknowledgements

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

3. References

3.1. Normative References

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

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

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

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

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

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

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

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

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

3.2. Informative References

[I-D.ietf-netconf-restconf]
Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", draft-ietf-netconf-restconf-07 (work in progress), July 2015.

[I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-19 (work in progress), May 2015.

Appendix A. Change log

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

A.1. Changes between versions -03 and -04

- Follow VRF-centric model
- IP single-hop session configuration in BFD clients

A.2. Changes between versions -02 and -03

- Fixed date mismatch
- Updated authors

A.3. Changes between versions -01 and -02

- Fixed errors and warnings from "pyang --ietf"
- Added appendix for "Change log"

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

In the YANG module section:

- Added missing filename
- 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

Santosh Pallagatti
Juniper Networks
India

Email: santoshpk@juniper.net

Mahesh Jethanandani
Cisco Systems

Email: mjethanandani@gmail.com

Greg Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 31, 2016

M. Jethanandani
Cisco Systems
A. Mishra
Ciena Corporation
A. Saxena
Citrix
M. Bhatia
Ionos Networks

September 28, 2015

Optimizing BFD Authentication
draft-mahesh-bfd-authentication-01

Abstract

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

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on March 31, 2016.

Copyright Notice

Table of Contents

1. Introduction

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

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

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

2. Authentication Mode

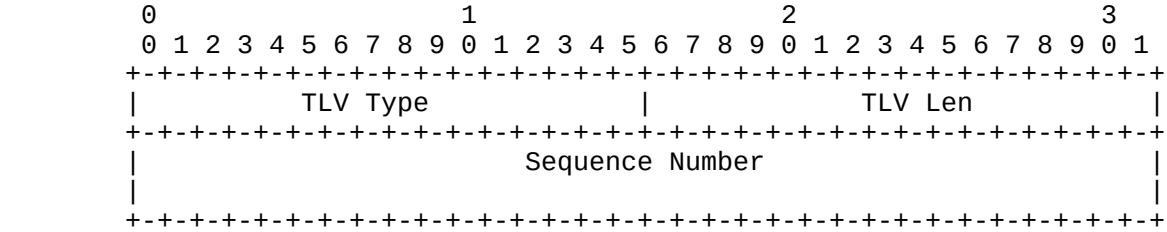
The cryptographic authentication mechanisms specified in BFD [RFC5880] describes enabling and disabling of authentication as a one time operation. As a security precaution, it mentions that authentication state be allowed to change at most once. Once enabled, every packet must have Authentication Bit set and the associated Authentication TLV appended. In addition, it states that an implementation SHOULD NOT allow the authentication state to be changed based on the receipt of a BFD Control packet.

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

Authenticated frames already carry the sequence number. The rest of the frames MUST contain the TLV specified in Section 3. This enables a monotonically increasing sequence number to be carried in each frame, and prevents man-in-the-middle from capturing and replaying the same frame again.

3. NULL Auth TLV

This section describes a new Authentication TLV as:



NULL Auth TLV

where:

TLV Type: The TLV Type. This field MUST be set to <IANA assigned>.

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

Sequence Number: is a monotonically increasing number while the session state is UP. Once the session goes down the Sequence number SHOULD be set to 0.

4. IANA Considerations

IANA is requested to assign a new Auth Type for the NULL Auth TLV.

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

5. Security Considerations

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

6. References

6.1. Normative References

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

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

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

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

[RFC6039] Manral, V., Bhatia, M., Jaeggli, J., and R. White, "Issues with Existing Cryptographic Protection Methods for Routing Protocols", RFC 6039, DOI 10.17487/RFC6039, October 2010, <http://www.rfc-editor.org/info/rfc6039>.

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

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

6.2. Informative References

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

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

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

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

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

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

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <http://www.rfc-editor.org/info/rfc2104>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <http://www.rfc-editor.org/info/rfc4086>.

[RFC4822] Atkinson, R. and M. Fanto, "RIPv2 Cryptographic Authentication", RFC 4822, DOI 10.17487/RFC4822, February 2007, <http://www.rfc-editor.org/info/rfc4822>.

[RFC5310] Bhatia, M., Manral, V., Li, T., Atkinson, R., White, R., and M. Fanto, "IS-IS Generic Cryptographic Authentication", RFC 5310, DOI 10.17487/RFC5310, February 2009, <http://www.rfc-editor.org/info/rfc5310>.

[RFC5709] Bhatia, M., Manral, V., Fanto, M., White, R., Barnes, M., Li, T., and R. Atkinson, "OSPFv2 HMAC-SHA Cryptographic Authentication", RFC 5709, DOI 10.17487/RFC5709, October 2009, <http://www.rfc-editor.org/info/rfc5709>.

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

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

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

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

Authors' Addresses

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

Phone: +1 (408) 526-8763
Email: mjethanandani@gmail.com

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

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

Ankur Saxena
Citrix
4988 Great America Pkwy
Santa Clara, CA 95054
USA

Email: ankurpsaxena@gmail.com

Manav Bhatia
Ionos Networks
Bangalore
India

Email: manav@ionosnetworks.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2016

S. Pallagatti, Ed.
B. Saji
S. Paragiri
Juniper Networks
V. Govindan
M. Mudigonda
Cisco
G. Mirsky
Ericsson

July 6, 2015

BFD for VXLAN
draft-spallagatti-bfd-vxlan-01

Abstract

This document describes use of Bidirectional Forwarding Detection (BFD) protocol for VXLAN . Comments on this draft should be directed to nvo3@ietf.org, rtg-bfd@ietf.org.

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 January 7, 2016.

Table of Contents

1. Introduction

"Virtual eXtensible Local Area Network (VXLAN)" has been defined in [RFC7348] that provides an encapsulation scheme which allows VM's to communicate in data centre network.

VXLAN is typically deployed in data centres on virtualized hosts, which may be spread across multiple racks. The individual racks may be parts of a different Layer 3 network or they could be in a single Layer 2 network. The VXLAN segments/overlay networks are overlaid on top of these Layer 2 or Layer 3 networks.

A VM can communicate with a VM in other host only if they are on same VXLAN. VM's are unaware of VXLAN tunnels as VXLAN tunnel terminates on VTEP (hypervisor/TOR). VTEP (hypervisor/TOR) are responsible for encapsulating and decapsulating frames sent from VM's.

Since underlay is a L3 network, connectivity check for these tunnels becomes important. BFD as defined in [RFC5880] can be used to monitor the VXLAN tunnels.

This draft addresses requirements outlined in [I-D.ashwood-nvo3-operational-requirement]. Specifically with reference to the OAM model to Figure 3 of [I-D.ashwood-nvo3-operational-requirement], this draft outlines proposal to implement the OAM mechanism between the NV Edges using BFD.

2. Use cases

Main use case of BFD for VXLAN is for tunnel connectivity check. There are other use cases such as

Layer 2 VM's:

Most deployments will have VM's with only L2 capabilities and may not understand L3. BFD being a L3 protocol can be used for tunnel connectivity check, where BFD will start and terminate at the NV Edge (VTEPs).

It is possible to aggregate the connectivity checks for multiple tenants by running a BFD session between the VTEPs over VxLAN tunnel. In rest of this document terms NV Edge and VTEP are used interchangeably.

Fault localization:

It is also possible that VM's are L3 aware and can possibly host a BFD session. In these cases BFD sessions can be established between VM's for connectivity check. In addition a BFD session can be established between VTEPs for tunnel connectivity check. Having a hierarchical OAM model helps localize faults.

Service node reachability:

Service node is responsible for sending BUM traffic. In case of service node tunnel terminates at VTEP and it might not even host VM's. If TOR's/Hypervisor wants to check service node reachability then it would like run BFD session over VXLAN tunnel to service node.

3. Deployment

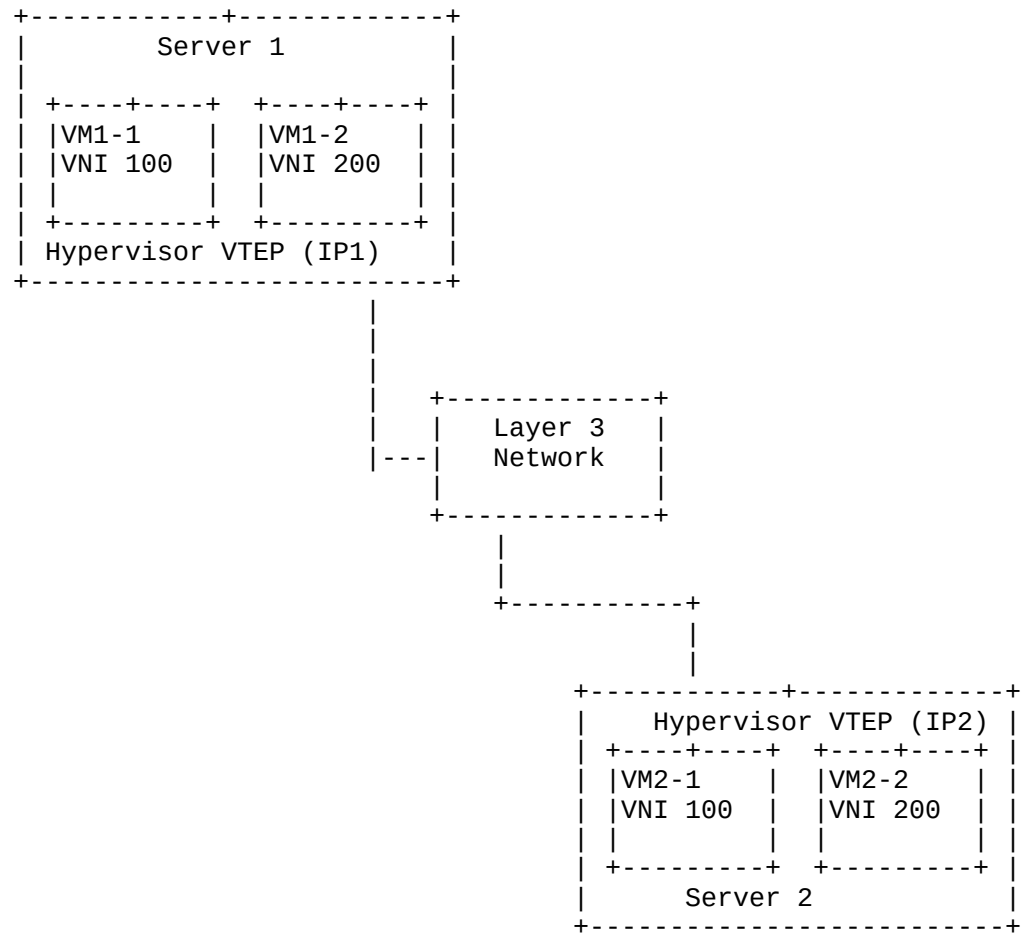


Figure 1

Figure 1 illustrates a scenario where we have two servers, each of them hosting two VMs. These VTEPs terminate two VXLAN tunnels with VNI number 100 and 200 between them. Separate BFD sessions can be established between the VTEPs (IP1 and IP2) for monitoring each of the VXLAN tunnels (VNI 100 and 200). No BFD packet intended to Hypervisor VTEP should be forwarded to VM's as VM's may drop this leading to false negative. This method is also applicable VTEP which are either software or physical device.

4. BFD Packet Encapsulation

VxLAN packet format has been defined in Section 5 of [RFC7348]. The Outer IP/UDP and VXLAN headers MUST be encoded by the sender as per [RFC7348].

If VTEP is equipped with GPE header capitalises and decides to use GPE instead of VXLAN then GPE header MUST be encoded as per Section 3.3 of [I-D.quinn-vxlan-gpe]. Next Protocol Field in GPE header MUST be set to IPv4 or IPv6.

Details of how VTEP decides to use VXLAN or GPE header is outside the scope of this document.

The BFD packet MUST be carried inside the inner MAC frame of the VxLAN packet. The inner MAC frame carrying the BFD payload has the following format:

Ethernet Header:

Destination MAC: This MUST be a well-known MAC [TBD] OR the MAC address of the destination VTEP. The details of how the destination MAC address is obtained is outside the scope of this document.

Source MAC: MAC address of the originating VTEP

IP header:

Source IP: IP address of the originating VTEP.

Destination IP: IP address of the terminating VTEP.

TTL: This MUST be set to 1. This is to ensure that the BFD packet is not routed within the L3 underlay network.

Note: Inner source and destination IP needs more discussion in WG.

The fields of the UDP header and the BFD control packet are encoded as specified in RFC 5881.

5. Reception of BFD packet

Once a packet is received, VTEP MUST validate the packet as described in Section 4.1 of [RFC7348]. If the Destination MAC of the inner MAC frame matches the well-known MAC or the MAC address of the VTEP the packet MUST be processed further.

The UDP destination port and the TTL of the inner MAC frame MUST be validated to determine if the received packet can be processed by BFD. BFD packet with inner MAC set to VTEP or well-known MAC address MUST not be forwarded to VM's.

5.1. Demux of the BFD packet

Demux of IP BFD packet has been defined in Section 3 of [RFC5881]. Since multiple BFD sessions may be running between two VTEPs, there needs to be a mechanism for demultiplexing received BFD packets to the proper session. The procedure for demultiplexing packets with Your Discriminator = 0 is different from [RFC5880]. For such packets, the BFD session is identified using the VNID, the source IP and the destination IP of the packet. If BFD packet is received with non-zero your discriminator then BFD session should be demultiplexed only with your discriminator as the key.

6. Echo BFD

Support for echo BFD is outside the scope of this document.

7. IANA Considerations

The well-known MAC to be used for the Destination MAC address of the inner MAC frame needs to be defined

8. Security Considerations

Document recommends setting of inner IP TTL to 1 which could lead to DDoS attack, implementation MUST have throttling in place. Throttling MAY be relaxed for BFD packeted based on port number.

Other than inner IP TTL set to 1 this specification does not raise any additional security issues beyond those of the specifications referred to in the list of normative references.

9. Contributors

Reshad Rahman
rrahman@cisco.com
Cisco

10. Acknowledgements

Authors would like to thank Jeff Hass of Juniper Networks for his reviews and feedback on this material.

11. Normative References

[I-D.ashwood-nvo3-operational-requirement]
Ashwood-Smith, P., Iyengar, R., Tsou, T., Sajassi, A., Boucadair, M., Jacquenet, C., and M. Daikoku, "NVO3 Operational Requirements", draft-ashwood-nvo3-operational-requirement-03 (work in progress), July 2013.

[I-D.ietf-bfd-seamless-base]
Akiya, N., Pignataro, C., Ward, D., Bhatia, M., and J. Networks, "Seamless Bidirectional Forwarding Detection (S-BFD)", draft-ietf-bfd-seamless-base-05 (work in progress), June 2015.

[I-D.quinn-vxlan-gpe]
Quinn, P., Manur, R., Kreeger, L., Lewis, D., Maino, F., Smith, M., Agarwal, P., Yong, L., Xu, X., Elzur, U., Garg, P., and D. Melman, "Generic Protocol Extension for VXLAN", draft-quinn-vxlan-gpe-04 (work in progress), February 2015.

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

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

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

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, August 2014.

Authors' Addresses

Santosh Pallagatti (editor)
Juniper Networks
Embassy Business Park
Bangalore, KA 560093
India

Email: santoshpk@juniper.net

Basil Saji
Juniper Networks
Embassy Business Park
Bangalore, KA 560093
India

Email: sbasil@juniper.net

Sudarsan Paragiri
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, California 94089-1206
USA

Email: sparagiri@juniper.net

Vengada Prasad Govindan
Cisco

Email: venggovi@cisco.com

Mallik Mudigonda
Cisco

Email: mmudigon@cisco.com

Greg Mirsky
Ericsson

Email: gregory.mirsky@ericsson.com