Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 22, 2016

L. Zheng
Huawei Technologies
S. Aldrin
Google
G. Zheng
Huawei Technologies
G. Mirsky
Ericsson
R. Rahman
Cisco Systems

March 21, 2016

Yang Data Model for LSP-PING
draft-zheng-mpls-lsp-ping-yang-cfg-03.txt

Abstract

When an LSP fails to deliver user traffic, the failure cannot always be detected by the MPLS control plane. RFC4379 defines a mechanism that would enable users to detect such failure and to isolate faults. 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 RFC[6241]. This document defines a YANG data model that can be used to configure and manage LSP-Ping.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 22, 2016.

Table of Contents

1. Introduction

When an LSP fails to deliver user traffic, the failure cannot always be detected by the MPLS control plane. [RFC4379] defines a mechanism that would enable users to detect such failure and to isolate faults. 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]. This document defines

a YANG data model that can be used to configure and manage LSP-Ping [RFC4379].

The rest of this document is organized as follows. Section 2 presents the scope of this document. Section 3 provides the design of the LSP-Ping configuration data model in details by containers. Section 4 presents the complete data hierarchy of LSP-Ping YANG model. Section 5 discusses the interaction between LSP-Ping data model and other MPLS tools data models. Section 6 specifies the YANG module and section 7 lists examples which conform to the YANG module specified in this document. Finally, security considerations are discussed in Section 8.

1.1. Support of Long Running Command with NETCONF

LSP Ping is one of examples of what can described as "long-running operation". Unlike most of configuration operations that result in single response execution of an LSP Ping triggers multiple responses from a node under control. The question of implementing long-running operation in NETCONF is still open and possible solutions being discussed:

1. Consecutive Remote Processing Calls (RPC) to poll for results;

2. Model presented in[RFC4560] ;

3. The one outlined in [I-D.mahesh-netconf-persistent].

The problem of long-running operation as well can be considered as a case of controlling and obtaining results from a Measurement Agent (MA) as defined in [I-D.ietf-lmap-framework].

1.2. Contributors

Yanfeng Zhang (Huawei Technologies) contributed to the definition of the YANG module in Section 6.

2. Scope

The fundamental mechanism of LSP-Ping is defined in [RFC4379]. The on-going work of [I-D.ietf-mpls-rfc4379bis] is to take LSP Ping to an Internet Standard, updates from [RFC5462] and [RFC6829] have been incorporated. These updates of LSP-Ping has been considered in this document. Extensions of LSP-Ping has been developed over the years. There are for example extensions for performing LSP ping over P2MP MPLS LSPs [RFC6425] and trace-route over MPLS tunnels [RFC6424] etc. These extensions will be considered in update of this document.

3. Design of the Data Model

This YANG data model is defined to be used to configure and manage LSP-Ping and it provides the following features:

1. Configuration of control information of a LSP-Ping test;

2. Configuration of schedule parameters of a LSP-Ping test;

3. Display of result information of a LSP-Ping test.

The top level container lsp-pings holds the configuration of control information, schedule parameters and result information for multi instances of LSP-Ping test.

3.1. Configuration of Control Information

Container lsp-pings:lsp-ping:control-info defines the configuration parameters which control a LSP-Ping test. Examples are the target-fec-type/target-fec of the echo request packet and the reply mode of the echo reply packet. Values of some parameters may be auto-assigned by the system, but in several cases there is a requirement for configuration of these parameters. Examples of such parameters are source address and outgoing interface.

The data hierarchy for control information configuration is presented below:

```
module: ietf-lspping
   +--rw lsp-pings
      +--rw lsp-ping* [lsp-ping-name]
         +--rw lsp-ping-name          string
         +--rw control-info
         |  +--rw target-fec-type?       target-fec-type
         |  +--rw (target-fec)?
         |  |  +--:(ip-prefix)
         |  |  |  +--rw ip-address?            inet:ip-address
         |  |  +--:(bgp)
         |  |  |  +--rw bgp?                   inet:ip-address
         |  |  +--:(rsvp)
         |  |  |  +--rw tunnel-interface?      uint32
         |  |  +--:(vpn)
         |  |  |  +--rw vrf-name?              uint32
         |  |  |  +--rw vpn-ip-address?        inet:ip-address
         |  |  +--:(pw)
         |  |  |  +--rw vcid?                  uint32
         |  |  +--:(vpls)
         |  |     +--rw vsi-name?              string
         |  +--rw traffic-class?         uint8
         |  +--rw reply-mode?            reply-mode
         |  +--rw timeout?               uint32
         |  +--rw timeout-units?         units
         |  +--rw interval?              uint32
         |  +--rw interval-units?        units
         |  +--rw probe-count?           uint32
         |  +--rw data-size?             uint32
         |  +--rw data-fill?             string
         |  +--rw description?           string
         |  +--rw source-address-type?   inet:ip-version
         |  +--rw source-address?        inet:ip-address
         |  +--rw ttl?                   uint32
         |  +--rw (outbound)?
         |     +--:(interface)
         |     |  +--rw interface-name?        string
         |     +--:(nexthop)
         |        +--rw nexthop?               inet:ip-address
```

3.2. Configuration of Schedule Parameters

Container lsp-pings:lsp-ping:schedule-parameters defines the schedule parameters of a LSP-Ping test, which basically describes when to start and when to end the test. Four start modes and three end modes are defined respectively. To be noted that, the configuration of "interval" and "probe-count" parameter defined in container lsp-pings:lsp-ping:control-info could also determine when the test ends

implicitly. All these three parameters are optional. If "interval" and "probe-count" are not configured by the user, the default values will be used by the system. If "end-test" is configured by the user, the actual end time of the LSP-Ping test is the smaller one between the configuration value of "end-test" and the time implicitly determined by the configuration value of "interval"/"probe-count".

The data hierarchy for schedule information configuration is presented below:

```
module: ietf-lspping
   +--rw lsp-pings
      +--rw lsp-ping* [lsp-ping-name]
         +--rw lsp-ping-name          string
         +--rw control-info
         ...
         +--rw schedule-parameters
         |  +--rw (start-test)?
         |  |  +--:(now)
         |  |  |  +--rw start-test-now?           empty
         |  |  +--:(at)
         |  |  |  +--rw start-test-at?            yang:date-and-time
         |  |  +--:(delay)
         |  |  |  +--rw start-test-delay?         uint32
         |  |  |  +--rw start-test-delay-units?   units
         |  |  +--:(daily)
         |  |     +--rw start-test-daily?         yang:date-and-time
         |  +--rw (end-test)?
         |     +--:(at)
         |     |  +--rw end-test-at?              yang:date-and-time
         |     +--:(delay)
         |     |  +--rw end-test-delay?           uint32
         |     |  +--rw end-test-delay-units?     units
         |     +--:(lifetime)
         |        +--rw end-test-lifetime?        uint32
         |        +--rw lifetime-units?           units
```

3.3. Display of Result Information

Container lsp-pings:lsp-ping:result-info shows the result of the current LSP-Ping test. Both the statistical result e.g. min-rtt, max rtt, and per test probe result e.g. return code, return subcode, are shown.

The data hierarchy for display of result information is presented below:

```
module: ietf-lspping
   +--rw lsp-pings
      +--rw lsp-ping* [lsp-ping-name]
         +--rw lsp-ping-name          string
         +--rw control-info
         ...
         +--rw schedule-parameters
         ...
         +--ro result-info
            +--ro operational-status?    operational-status
            +--ro source-address-type?   inet:ip-version
            +--ro source-address?        inet:ip-address
            +--ro target-fec-type?       target-fec-type
            +--ro (target-fec)?
            |  +--:(ip-prefix)
            |  |  +--ro ip-address?            inet:ip-address
            |  +--:(bgp)
            |  |  +--ro bgp?                   inet:ip-address
            |  +--:(rsvp)
            |  |  +--ro tunnel-interface?      uint32
            |  +--:(vpn)
            |  |  +--ro vrf-name?              uint32
            |  |  +--ro vpn-ip-address?        inet:ip-address
            |  +--:(pw)
            |  |  +--ro vcid?                  uint32
            |  +--:(vpls)
            |     +--ro vsi-name?              string
            +--ro min-rtt?               uint32
            +--ro max-rtt?               uint32
            +--ro average-rtt?           uint32
            +--ro probe-responses?       uint32
            +--ro sent-probes?           uint32
            +--ro sum-of-squares?        uint32
            +--ro last-good-probe?       yang:date-and-time
            +--ro probe-results
               +--ro probe-result* [probe-index]
                  +--ro probe-index        uint32
                  +--ro return-code?       uint8
                  +--ro return-sub-code?   uint8
                  +--ro rtt?               uint32
                  +--ro result-type?       result-type
```

4. Data Hierarchy

The complete data hierarchy of LSP-Ping YANG model is presented below.

```
   module: ietf-lspping
      +--rw lsp-pings
         +--rw lsp-ping* [lsp-ping-name]
            +--rw lsp-ping-name          string
            +--rw control-info
            |  +--rw target-fec-type?       target-fec-type
            |  +--rw (target-fec)?
            |  |  +--:(ip-prefix)
            |  |  |  +--rw ip-address?            inet:ip-address
            |  |  +--:(bgp)
            |  |  |  +--rw bgp?                   inet:ip-address
            |  |  +--:(rsvp)
            |  |  |  +--rw tunnel-interface?      uint32
            |  |  +--:(vpn)
            |  |  |  +--rw vrf-name?              uint32
            |  |  |  +--rw vpn-ip-address?        inet:ip-address
            |  |  +--:(pw)
            |  |  |  +--rw vcid?                  uint32
            |  |  +--:(vpls)
            |  |     +--rw vsi-name?              string
            |  +--rw traffic-class?         uint8
            |  +--rw reply-mode?            reply-mode
            |  +--rw timeout?               uint32
            |  +--rw timeout-units?         units
            |  +--rw interval?              uint32
            |  +--rw interval-units?        units
            |  +--rw probe-count?           uint32
            |  +--rw data-size?             uint32
            |  +--rw data-fill?             string
            |  +--rw description?           string
            |  +--rw source-address-type?   inet:ip-version
            |  +--rw source-address?        inet:ip-address
            |  +--rw ttl?                   uint32
            |  +--rw (outbound)?
            |     +--:(interface)
            |     |  +--rw interface-name?        string
            |     +--:(nexthop)
            |        +--rw nexthop?               inet:ip-address
            +--rw schedule-parameters
            |  +--rw (start-test)?
            |  |  +--:(now)
            |  |  |  +--rw start-test-now?           empty
            |  |  +--:(at)
            |  |  |  +--rw start-test-at?            yang:date-and-time
            |  |  +--:(delay)
            |  |  |  +--rw start-test-delay?         uint32
            |  |  |  +--rw start-test-delay-units?   units
            |  |  +--:(daily)
```

```
|  |     +--rw start-test-daily?         yang:date-and-time
|  +--rw (end-test)?
|     +--:(at)
|     |  +--rw end-test-at?              yang:date-and-time
|     +--:(delay)
|     |  +--rw end-test-delay?           uint32
|     |  +--rw end-test-delay-units?     units
|     +--:(lifetime)
|        +--rw end-test-lifetime?        uint32
|        +--rw lifetime-units?           units
+--ro result-info
   +--ro operational-status?    operational-status
   +--ro source-address-type?   inet:ip-version
   +--ro source-address?        inet:ip-address
   +--ro target-fec-type?       target-fec-type
   +--ro (target-fec)?
   |  +--:(ip-prefix)
   |  |  +--ro ip-address?            inet:ip-address
   |  +--:(bgp)
   |  |  +--ro bgp?                   inet:ip-address
   |  +--:(rsvp)
   |  |  +--ro tunnel-interface?      uint32
   |  +--:(vpn)
   |  |  +--ro vrf-name?              uint32
   |  |  +--ro vpn-ip-address?        inet:ip-address
   |  +--:(pw)
   |  |  +--ro vcid?                  uint32
   |  +--:(vpls)
   |     +--ro vsi-name?              string
   +--ro min-rtt?               uint32
   +--ro max-rtt?               uint32
   +--ro average-rtt?           uint32
   +--ro probe-responses?       uint32
   +--ro sent-probes?           uint32
   +--ro sum-of-squares?        uint32
   +--ro last-good-probe?       yang:date-and-time
   +--ro probe-results
      +--ro probe-result* [probe-index]
         +--ro probe-index        uint32
         +--ro return-code?       uint8
         +--ro return-sub-code?   uint8
         +--ro rtt?               uint32
         +--ro result-type?       result-type
```

5. Interaction with other MPLS OAM Tools Models

TBA

6. LSP-Ping Yang Module

```
<CODE BEGINS> file "ietf-lspping@2015-07-02.yang"
module ietf-lspping {
  namespace "urn:ietf:params:xml:ns:yang:ietf-lspping";
  //namespace need to be assigned by IANA
  prefix "lspping";

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

  organization "IETF Multiprotocl Label Switching Working Group";
  contact "draft-zheng-mpls-lsp-ping-yang-cfg";
  description "MPLS LSP-PING Yang Module";
  revision "2016-03-18" {
    description "03 version, refine the target fec type,
             as per RFC4379; updates per draft-ietf-mpls-rfc4379bis";
    reference "draft-zheng-mpls-lsp-ping-yang-cfg";
  }

  typedef target-fec-type {
    type enumeration {
      enum ip-prefix {
        value "0";
        description "IPv4/IPv6 prefix";
      }
      enum bgp {
        value "1";
        description "BGP IPv4/IPv6 prefix";
      }
      enum rsvp {
        value "2";
        description "Tunnel interface";
      }
      enum vpn {
        value "3";
        description "VPN IPv4/IPv6 prefix";
      }
      enum pw {
        value "4";
```

```
          description "FEC 128 pseudowire IPv4/IPv6";
        }
        enum vpls {
          value "5";
          description "FEC 129 pseudowire IPv4/IPv6";
        }
      }
      description "Target FEC type.";
    }

    typedef reply-mode {
      type enumeration {
        enum do-not-reply {
          value "1";
          description "Do not reply";
        }
        enum reply-via-udp {
          value "2";
          description "Reply via an IPv4/IPv6 UDP packet";
        }
        enum reply-via-udp-router-alert {
          value "3";
          description "Reply via an IPv4/IPv6 UDP packet with
          Router Alert";
        }
        enum reply-via-control-channel {
          value "4";
          description "Reply via application level control
          channel";
        }
      }
      description "Reply mode.";
    }

    typedef units  {
      type enumeration {
        enum seconds {
          description "Seconds";
        }
        enum milliseconds {
          description "Milliseconds";
        }
        enum microseconds {
          description "Microseconds";
        }
        enum nanoseconds {
          description "Nanoseconds";
        }
```

```
      }
      description "Time units";
    }

    typedef operational-status {
      type enumeration {
        enum enabled {
          value "1";
          description "The Test is active.";
        }
        enum disabled {
          value "2";
          description "The test has stopped.";
        }
        enum completed {
          value "3";
          description "The test is completed.";
        }
      }
      description "Operational state of a LSP Ping test.";
    }

    typedef result-type {
      type enumeration {
        enum success {
          value "1";
          description "The test probe is successed.";
        }
        enum fail {
          value "2";
          description "The test probe has failed.";
        }
        enum timeout {
          value "3";
          description "The test probe is timeout.";
        }
      }
      description "Result of each LSP Ping test probe.";
    }

    container lsp-pings {
      description "Multi instance of LSP Ping test.";
      list lsp-ping {
        key "lsp-ping-name";
        description "LSP Ping test";
        leaf lsp-ping-name {
          type string {
              length "1..31";
```

```
          }
          mandatory "true";
          description "LSP Ping test name.";
        }
        container control-info {
          description "Control information of the LSP Ping test.";
          leaf target-fec-type {
            type target-fec-type;
            description "Specifies the address type of Target FEC.";
          }
          choice target-fec {
            case ip-prefix {
              leaf ip-address {
                type inet:ip-address;
                description "IPv4/IPv6 Prefix.";
              }
            }
            case bgp {
              leaf bgp {
                type inet:ip-address;
                description "BGP IPv4/IPv6 Prefix.";
              }
            }
            case rsvp {
              leaf tunnel-interface {
                type uint32;
                description "Tunnel interface";
              }
            }
            case vpn {
              leaf vrf-name {
                type uint32;
                description "Layer3 VPN Name.";
              }
              leaf vpn-ip-address {
                type inet:ip-address;
                description "Layer3 VPN IPv4 Prefix.";
              }
            }
            case pw {
              leaf vcid {
                type uint32;
                description "VC ID";
              }
            }
            case vpls {
              leaf vsi-name {
                type string;
```

```
            description "VPLS VSI";
          }
        }
        description "Specifies the address of Target FEC";
      }
      leaf traffic-class {
        type uint8;
        description "Specifies the Traffic Class.";
      }
      leaf reply-mode {
        type reply-mode;
        description "Specifies the reply mode.";
      }
      leaf timeout {
        type uint32;
        description "Specifies the time-out value for a
        LSP Ping operation.";
      }
      leaf timeout-units {
        type units;
        description "Time-out units.";
      }
      leaf interval {
        type uint32;
        default 1;
        description "Specifies the interval to send a LSP Ping
        echo request packet(probe) as part of one LSP Ping test.";
      }
      leaf interval-units {
        type units;
        default seconds;
        description "Interval units.";
      }
      leaf probe-count {
        type uint32;
        default 5;
        description "Specifies the number of probe sent of one
        LSP Ping test.";
      }
      leaf data-size {
        type uint32;
        description "Specifies the size of the data portion to
        be transmitted in a LSP Ping operation, in octets.";
      }
      leaf data-fill {
        type string{
            length "0..1564";
        }
```

```
            description "Used together with the corresponding
            data-size value to determine how to fill the data
            portion of a probe packet.";
          }
          leaf description {
            type string{
                length "1..31";
            }
            description "A descriptive name of the LSP Ping test.";
          }
          leaf source-address-type {
            type inet:ip-version;
            description "Specifies the type of the source address.";
          }
          leaf source-address {
            type inet:ip-address;
            description "Specifies the source address.";
          }
          leaf ttl {
            type uint32;
            default 255;
            description "Time to live.";
          }
          choice outbound {
            case interface {
              leaf interface-name{
                type string{
                  length "1..255";
                }
                description "Specifies the outgoing interface.";
              }
            }
            case nexthop{
              leaf nexthop {
                type inet:ip-address;
                description "Specifies the nexthop.";
              }
            }
            description "Specifies the out interface or nexthop";
          }
        }

        container schedule-parameters {
          description "LSP Ping test schedule parameter";
          choice start-test{
            case now {
              leaf start-test-now {
                type empty;
```

```
                description "Start test now.";
              }
            }
            case at {
              leaf start-test-at {
                type yang:date-and-time;
                description "Start test at a specific time.";
              }
            }
            case delay {
              leaf start-test-delay {
                type uint32;
                description "Start after a specific delay.";
              }
              leaf start-test-delay-units {
                type units;
                default seconds;
                description "Delay units.";
              }
            }
            case daily {
              leaf start-test-daily {
                type yang:date-and-time;
                description "Start test daily.";
              }
            }
            description "Specifies when the test begins to start,
            include 4 schedule method: start now(1), start at(2),
            start delay(3), start daily(4).";
          }

          choice end-test{
            case at {
              leaf end-test-at{
                type yang:date-and-time;
                description "End test at a specific time.";
              }
            }
            case delay {
              leaf end-test-delay {
                type uint32;
                description "End after a specific delay.";
              }
              leaf end-test-delay-units {
                type units;
                default seconds;
                description "Delay units.";
              }
```

```
            }
            case lifetime {
              leaf end-test-lifetime {
                type uint32;
                description "Set the test lifetime.";
              }
              leaf lifetime-units {
                type units;
                default seconds;
                description "Lifetime units.";
              }
            }
            description "Specifies when the test ends, include 3
            schedule method: end at(1), end delay(2),
            end lifetime(3).";
          }
        }

        container result-info {
          config "false";
          description "LSP Ping test result information.";
          leaf operational-status {
            type operational-status;
            description "Operational state of a LSP Ping test";
          }
          leaf source-address-type {
            type inet:ip-version;
            description "The source address type.";
          }
          leaf source-address {
            type inet:ip-address;
            description "The source address of the test.";
          }
          leaf target-fec-type {
            type target-fec-type;
            description "The Target FEC address type.";
          }
          choice target-fec {
            case ip-prefix {
              leaf ip-address {
                type inet:ip-address;
                description "IPv4/IPv6 Prefix.";
              }
            }
            case bgp {
              leaf bgp {
                type inet:ip-address;
                description "BGP IPv4/IPv6 Prefix.";
```

```
            }
          }
          case rsvp {
            leaf tunnel-interface {
              type uint32;
              description "Tunnel interface";
            }
          }
          case vpn {
            leaf vrf-name {
              type uint32;
              description "Layer3 VPN Name.";
            }
            leaf vpn-ip-address {
              type inet:ip-address;
              description "Layer3 VPN IPv4 Prefix.";
            }
          }
          case pw {
            leaf vcid {
              type uint32;
              description "VC ID";
            }
          }
          case vpls {
            leaf vsi-name {
              type string;
              description "VPLS VSI";
            }
          }
          description "The Target FEC address";
        }
        leaf min-rtt {
          type uint32;
          description "The minimum LSP Ping round-trip-time (RTT)
          received.";
        }
        leaf max-rtt {
          type uint32;
          description "The maximum LSP Ping round-trip-time (RTT)
          received.";
        }
        leaf average-rtt {
          type uint32;
          description "The current average LSP Ping round-trip-time
          (RTT).";
        }
        leaf probe-responses {
```

```
            type uint32;
            description "Number of responses received for the
            corresponding LSP Ping test.";
          }
          leaf sent-probes {
            type uint32;
            description "Number of probes sent for the
            corresponding LSP Ping test.";
          }
          leaf sum-of-squares {
            type uint32;
            description "The sum of the squares for all
            replys received.";
          }
          leaf last-good-probe {
            type yang:date-and-time;
            description "Date and time when the last response
            was received for a probe.";
          }

          container probe-results {
            description "Result info of test probes.";
            list probe-result {
              key "probe-index";
              description "Result info of each test probe.";
              leaf probe-index {
                type uint32;
                description "Probe index";
              }
              leaf return-code {
                type uint8;
                description "The Return Code set in the echo reply.";
              }
              leaf return-sub-code {
                 type uint8;
                 description "The Return Sub-code set in the
                 echo reply.";
              }
              leaf rtt {
                type uint32;
                description "The round-trip-time (RTT) received.";
              }
              leaf result-type {
                type result-type;
                description "The probe result type.";
              }
            }
          }
```

```
        }
      }
    }
  }
  <CODE ENDS>
```

7. Examples

The following examples shows the netconf RPC communication between client and server for one LSP-Ping test case.

7.1. Configuration of Control Information

Configure the control-info for sample-test-case.

```
Request from netconf client:
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <lsp-pings xmlns="urn:ietf:params:xml:ns:yang:ietf-lspping">
        <lsp-ping>
          <lsp-ping-name>sample-test-case</lsp-ping-name>
          <control-info>
            <target-fec-type>ip-prefix</target-fec-type>
            <ip-prefix>112.80.248.74</ip-prefix>
            <reply-mode>reply-via-udp</reply-mode>
            <timeout>1</timeout>
            <timeout-units>seconds</timeout-units>
            <interval>1</interval>
            <interval-units>seconds</interval-units>
            <probe-count>6</probe-count>
            <admin-status>enabled</admin-status>
            <data-size>64</data-size>
            <data-fill>this is a lsp ping test</data-fill>
            <source-address-type>ipv4</source-address-type>
            <source-address>112.90.83.122</source-address>
            <ttl>56</ttl>
          </control-info>
        </lsp-ping>
      </lsp-pings>
    </config>
  </edit-config>
</rpc>

Reply from netconf server:
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

7.2. Configuration of Schedule Parameters

Set the schedule-parameters for sample-test-case to start the test.

```
Request from netconf client:
<rpc message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <lsp-pings xmlns="urn:ietf:params:xml:ns:yang:ietf-lspping">
        <lsp-ping>
          <lsp-ping-name>sample-test-case</lsp-ping-name>
          <schedule-parameters>
            <start-test-now/>
          </schedule-parameters>
        </lsp-ping>
      </lsp-pings>
    </config>
  </edit-config>
</rpc>

Reply from netconf server:
<rpc-reply message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

7.3. Display of Result Information

Get the result-info of sample-test-case.

```
Request from netconf client:
<rpc message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <lsp-pings xmlns="urn:ietf:params:xml:ns:yang:ietf-lspping">
       <lsp-ping>
         <lsp-ping-name>sample-test-case</lsp-ping-name>
           <result-info/>
       </lsp-ping>
      </lsp-pings>
    </filter>
    </get>
</rpc>

Reply from netconf server:
<rpc-reply message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <lsp-pings xmlns="urn:ietf:params:xml:ns:yang:ietf-lspping">
      <lsp-ping>
        <lsp-ping-name>sample-test-case</lsp-ping-name>
```

```
        <result-info>
          <operational-status>completed</operational-status>
          <source-address-type>ipv4</source-address-type>
          <source-address>112.90.83.122</source-address>
          <target-fec-type>ip-prefix</target-fec-type>
          <ip-prefix>112.80.248.74</ip-prefix>
          <min-rtt>10</min-rtt>
          <max-rtt>56</max-rtt>
          <average-rtt>36</average-rtt>
          <probe-responses>6</probe-responses>
          <sent-probes>6</sent-probes>
          <sum-of-squares>8882</sum-of-squares>
          <last-good-probe>2015-07-01T10:36:56<last-good-probe>
          <probe-results>
            <probe-result>
              <probe-index>0</probe-index>
              <return-code>0</return-code>
              <return-sub-code>3</return-sub-code>
              <rtt>10</rtt>
              <result-type>success</result-type>
            </probe-result>
            <probe-result>
              <probe-index>1</probe-index>
              <return-code>0</return-code>
              <return-sub-code>3</return-sub-code>
              <rtt>56</rtt>
              <result-type>success</result-type>
            </probe-result>
            <probe-result>
              <probe-index>2</probe-index>
              <return-code>0</return-code>
              <return-sub-code>3</return-sub-code>
              <rtt>35</rtt>
              <result-type>success</result-type>
            </probe-result>
            <probe-result>
              <probe-index>3</probe-index>
              <return-code>0</return-code>
              <return-sub-code>3</return-sub-code>
              <rtt>38</rtt>
              <result-type>success</result-type>
            </probe-result>
            <probe-result>
              <probe-index>4</probe-index>
              <return-code>0</return-code>
              <return-sub-code>3</return-sub-code>
              <rtt>36</rtt>
              <result-type>success</result-type>
```

```
            </probe-result>
            <probe-result>
              <probe-index>5</probe-index>
              <return-code>0</return-code>
              <return-sub-code>3</return-sub-code>
              <rtt>41</rtt>
              <result-type>success</result-type>
            </probe-result>
          </probe-results>
        </result-info>
      </lsp-ping>
    </lsp-pings>
  </data>
</rpc-reply>
```

8. Security Considerations

The configuration and state data defined in this document 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 authors recommend to implement the NETCONF access control model [RFC6536] to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of config true nodes defined in the YANG module which are writable/creatable/deletable. These data nodes may be considered sensitive or vulnerable in some network environments. Write operations to these data nodes without proper protection can have a negative effect on network operations.

9. IANA Considerations

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

URI:TBA

10. Acknowledgements

We would also like to thank XXX.

11. References

11.1. Normative References

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

[RFC4379] Kompella, K. and G. Swallow, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", RFC 4379, DOI 10.17487/RFC4379, February 2006, <http://www.rfc-editor.org/info/rfc4379>.

[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>.

11.2. Informative References

[I-D.ietf-lmap-framework] Eardley, P., Morton, A., Bagnulo, M., Burbridge, T., Aitken, P., and A. Akhter, "A framework for Large-Scale Measurement of Broadband Performance (LMAP)", draft-ietf-lmap-framework-14 (work in progress), April 2015.

[I-D.ietf-mpls-rfc4379bis] Kompella, K., Pignataro, C., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures", draft-ietf-mpls-rfc4379bis-00 (work in progress), January 2016.

[I-D.mahesh-netconf-persistent] Jethanandani, M., "NETCONF and persistent responses", draft-mahesh-netconf-persistent-00 (work in progress), October 2014.

[RFC4560] Quittek, J., Ed. and K. White, Ed., "Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations", RFC 4560, DOI 10.17487/RFC4560, June 2006, <http://www.rfc-editor.org/info/rfc4560>.

[RFC5462] Andersson, L. and R. Asati, "Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field", RFC 5462, DOI 10.17487/RFC5462, February 2009, <http://www.rfc-editor.org/info/rfc5462>.

[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>.

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

[RFC6424] Bahadur, N., Kompella, K., and G. Swallow, "Mechanism for Performing Label Switched Path Ping (LSP Ping) over MPLS Tunnels", RFC 6424, DOI 10.17487/RFC6424, November 2011, <http://www.rfc-editor.org/info/rfc6424>.

[RFC6425] Saxena, S., Ed., Swallow, G., Ali, Z., Farrel, A., Yasukawa, S., and T. Nadeau, "Detecting Data-Plane Failures in Point-to-Multipoint MPLS - Extensions to LSP Ping", RFC 6425, DOI 10.17487/RFC6425, November 2011, <http://www.rfc-editor.org/info/rfc6425>.

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

[RFC6829] Chen, M., Pan, P., Pignataro, C., and R. Asati, "Label Switched Path (LSP) Ping for Pseudowire Forwarding Equivalence Classes (FECs) Advertised over IPv6", RFC 6829, DOI 10.17487/RFC6829, January 2013, <http://www.rfc-editor.org/info/rfc6829>.

Authors' Addresses

Lianshu Zheng
Huawei Technologies
China

Email: vero.zheng@huawei.com

Sam K. Aldrin
Google
USA

Email: aldrin.ietf@gmail.com

Guangying Zheng
Huawei Technologies
China

Email: zhengguangying@huawei.com

Greg Mirsky
Ericsson
USA

Email: gregory.mirsky@ericsson.com

Reshad Rahman
Cisco Systems
Canada

Email: rrahman@cisco.com