Nokia Service Router Linux

Release 24.7

OAM and Diagnostics Guide

3HE 20873 AAAA TQZZA
Edition: 01
July 2024

Nokia is committed to diversity and inclusion. We are continuously reviewing our customer documentation and consulting with standards bodies to ensure that terminology is inclusive and aligned with the industry. Our future customer documentation will be updated accordingly.

Table of contents

1 About this guide

This guide describes how to configure features such as mirroring and sFlow, and how to use the Operations, Administration, and Maintenance (OAM) and diagnostics tools with the Nokia Service Router Linux (SR Linux).

This document is intended for network technicians, administrators, operators, service providers, and others who need to understand how the router is configured.

Note:
This manual covers the current release and may also contain some content that will be released in later maintenance loads. See the *SR Linux Release Notes* for information about features supported in each load.

Configuration and command outputs shown in this guide are examples only; actual displays may differ depending on supported functionality and user configuration.

1.1 Precautionary and information messages

The following are information symbols used in the documentation.

DANGER: Danger warns that the described activity or situation may result in serious personal injury or death. An electric shock hazard could exist. Before you begin work on this equipment, be aware of hazards involving electrical circuitry, be familiar with networking environments, and implement accident prevention procedures.

WARNING: Warning indicates that the described activity or situation may, or will, cause equipment damage, serious performance problems, or loss of data.

Caution: Caution indicates that the described activity or situation may reduce your component or system performance.

Note: Note provides additional operational information.

Tip: Tip provides suggestions for use or best practices.

1.2 Conventions

Nokia SR Linux documentation uses the following command conventions.

- **Bold** type indicates a command that the user must enter.
- Input and output examples are displayed in `Courier` text.
- An open right-angle bracket indicates a progression of menu choices or simple command sequence (often selected from a user interface). Example: **start** > **connect to**.

- A vertical bar (|) indicates a mutually exclusive argument.
- Square brackets ([ ]) indicate optional elements.
- Braces ({ }) indicate a required choice. When braces are contained within square brackets, they indicate a required choice within an optional element.
- *Italic* type indicates a variable.

Generic IP addresses are used in examples. Replace these with the appropriate IP addresses used in the system.

2 What's new

This is the first release of this document.

3 Mirroring

Mirroring copies IPv4 and IPv6 packets seen on a specified source, such as an interface (port) or subinterface (VLAN), or matching an ACL entry, and sends the packets to a specific destination, such as a locally attached traffic analyzer or a tunnel toward a remote destination.

By default, the mirrored packets include IPv4/IPv6 headers, as well as Ethernet headers. Traffic from multiple sources can be mirrored to a single destination, although traffic from a specific source cannot be mirrored to multiple destinations.

3.1 Mirror sources

The source for mirrored traffic can be an interface or subinterface or an ACL filter.

- Interfaces / subinterfaces

  A mirror source can be an interface, including all subinterfaces within that interface. The source can be a single interface (for example, `interface ethernet-1/1`) or a LAG (for example, `interface lag1`). Either a LAG member or LAG port can be mirrored. When a LAG port is configured as a mirror source, mirroring is enabled on all ports making up the LAG.

  The source can be a specific VLAN; that is, a subinterface within an interface where VLAN tagging is enabled (for example, `interface ethernet-1/1.1` or `lag1.1`).

  You can configure mirroring for traffic in a specific direction (ingress only, egress only) or bidirectional traffic (both ingress and egress).

  It is possible for the mirror source operational state to be down because of resource exhaustion. If the mirror source is not mirroring packets, check the operational state of the mirror sources .

- ACL filters

  A mirror source can be an IPv4 or IPv6 ACL filter, applied under one or more interfaces or subinterfaces. Traffic matching entries in the ingress ACL filter (regardless of whether the action is accept or drop), can be mirrored to the destination.

The following table lists hardware platform support for each mirror source.

Table 1: Hardware applicability (source mirroring)

Source	7220 IXR-D2/D3	7220 IXR-D2L/D3L	7220 IXR-D4/D5	7250 IXR-6e/10e	7250 IXR-X3b
Interface (ingress)	Yes	Yes	Yes	Yes	Yes (LAG not supported)
Interface (egress)	Yes	Yes	Yes	Yes	Yes (LAG not supported)
Subinterface (ingress)	Yes	Yes	Yes	Yes	Yes (LAG not supported)

Source	7220 IXR-D2/D3	7220 IXR-D2L/D3L	7220 IXR-D4/D5	7250 IXR-6e/10e	7250 IXR-X3b
Subinterface (egress)	Yes	Yes	No	Yes	Yes (LAG not supported)
ACL filter (ingress)	Yes	Yes	Yes	Yes	Yes
ACL filter (egress)	No	No	No	No	No

Note: LAG not supported means the packets cannot be mirrored if the source is a LAG.

3.2 Mirror destinations

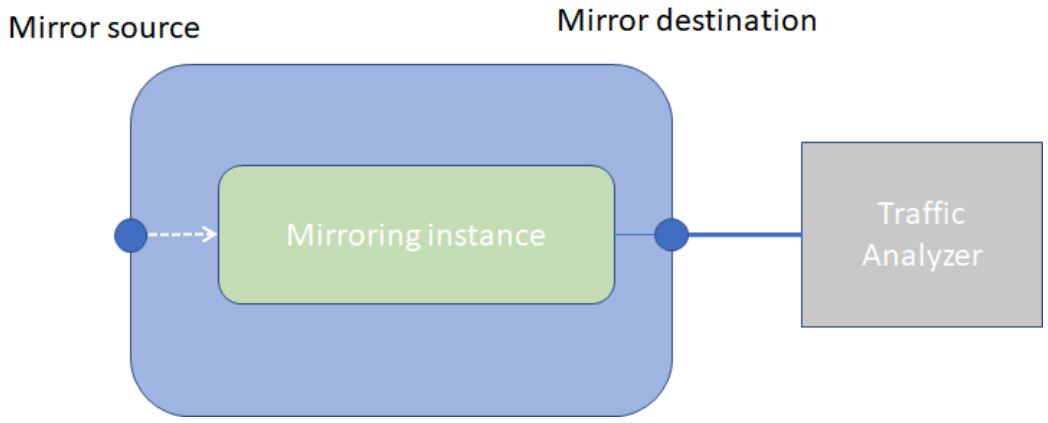
Traffic from the mirror source can be copied to a local destination (local mirroring) or encapsulated into a tunnel to a remote destination (remote mirroring).

Local mirroring

In a local mirroring configuration, both the mirror source and mirror destination reside on the same SR Linux node, as shown in Figure 1: Local mirroring.

In this configuration, the local destination is a Switched Port Analyzer (SPAN).

Figure 1: Local mirroring



For local mirroring, the following hardware types are supported:

- 7220 IXR-D2/D2L/D3/D3L
- 7220 IXR-D4/D5
- 7250 IXR-6e/10e
- 7250 IXR-X3b

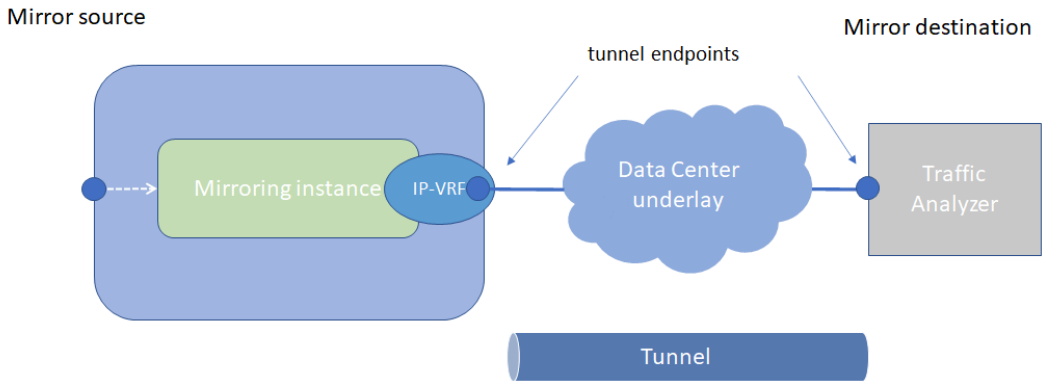
Remote mirroring

In a remote mirroring configuration, the mirror source and mirror destination are on different nodes. The mirror source resides on the SR Linux node, and the mirrored packets are encapsulated into a tunnel toward the mirror destination.

Figure 2: Remote mirroring shows a remote mirroring configuration. In this configuration, the remote destination is an Encapsulated Remote Switched Port Analyzer (ERSPAN).

Tunnel endpoints are defined within a specific network-instance, where the local tunnel endpoint IP address can be either a loopback subinterface address or any subinterface address within that network-instance.

Figure 2: Remote mirroring

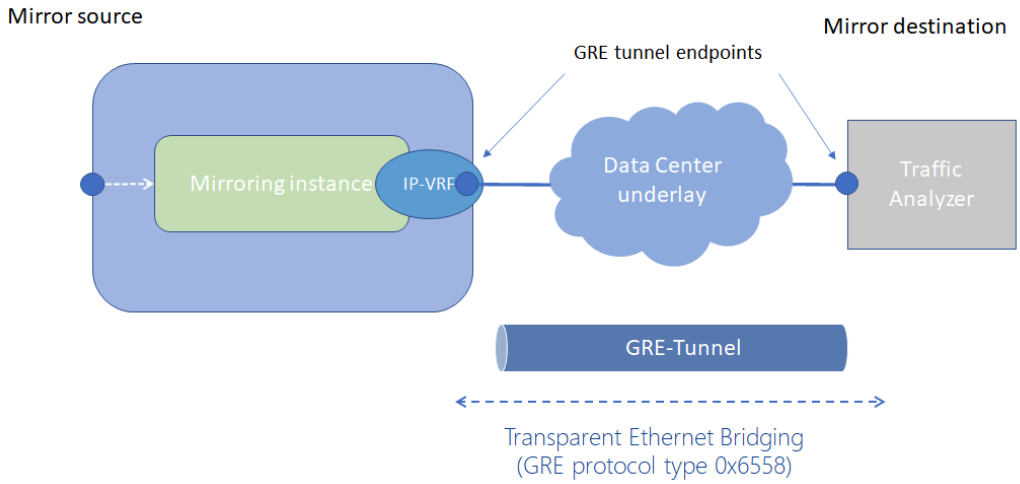


Mirroring to underlay (ERSPAN+GRE)

In a mirroring-to-underlay (ERSPAN+GRE) configuration, the mirrored packets, including IPv4/IPv6 header as well as Ethernet header, are tunneled using Transparent Ethernet Bridging (GRE protocol type 0x6558) or L3oGRE (protocol type 0x88be for 7250 IXR-6e/10e and 7250 IXR-X3b platforms only) toward the remote destination.

Figure 3: Mirroring to underlay shows a mirroring-to-underlay configuration.

Figure 3: Mirroring to underlay



For remote mirroring, the following hardware types are supported.

Table 2: Hardware applicability (destination mirroring - remote)

Destination	7220 IXR-D2/D3	7220 IXR-D2L/D3L	7220 IXR-D4/D5	7250 IXR-6e/10e	7250 IXR-X3b
Underlay destination (GRE +ERSPAN) - IPv4 (ingress and egress	Yes	Yes	Yes	Yes	Yes
Underlay destination (GRE +ERSPAN) - IPv6 (ingress-direction mirroring)	Yes	Yes	Yes	Yes	Yes
Underlay destination (GRE +ERSPAN) - IPv6 (egress-direction mirroring)	No	No	Yes	Yes	Yes

Note: For IXR-6e/10e and IXR-X systems, the remote encapsulation is L3oGRE. For all other platforms, the encapsulation is L2oGRE.

3.3 Configuring mirroring

To configure mirroring, you configure a mirroring-instance, which specifies the source and destination for the mirrored traffic. Multiple mirror sources can have a single destination, although traffic from a specific source cannot be mirrored to multiple destinations. Only one mirror destination can be configured per mirroring-instance. A mirror destination cannot be reused in multiple mirroring instances.

Within a mirroring-instance, if an interface is configured as mirror source, a subinterface within that interface cannot be added as another mirror source. If a LAG is defined as mirror destination, only the

first 8 members of the LAG carry mirrored traffic. Note that on 7220 IXR-D4 and D5 platforms, a mirror destination port cannot be a LAG.

Mirrored traffic is considered Best Effort (BE) Forwarding Class.

3.3.1 Configuring mirroring sources

Procedure

To configure mirroring, you specify the source and destination for mirrored traffic within a mirroring-instance. The source in a mirroring-instance can be traffic on a specified interface, subinterface, or LAG, or can be packets matching an ACL entry.

Example: interface source

The following example shows a mirroring-instance configuration with an interface as the source for mirrored traffic:

```
--{ * candidate shared default }--[  ]--
# info system mirroring
    system {
        mirroring {
            mirroring-instance 1 {
                admin-state enable
                mirror-source {
                    interface ethernet-1/5 {
                        direction ingress-egress
                    }
                }
            }
        }
    }
```

Example: ACL source

The following example configures an ACL with an entry that matches TCP packets and applies the ACL to a subinterface. A mirroring-instance is configured that uses packets matching the ACL as the source for mirrored traffic.

```
--{ * candidate shared default }--[  ]--
# info acl acl-filter ip_tcp type ipv4
    acl {
        acl-filter ip_tcp type ipv4 {
            entry 1000 {
                description Match_TCP_Protocol
                match {
                    ipv4 {
                        protocol tcp
                    }
                }
                action {
                    accept {
                    }
                }
            }
        }
```

```
    }
```

```
--{ +* candidate shared default }--[  ]--
# info acl interface ethernet-1/1.1
    acl {
        interface ethernet-1/1.1 {
            interface-ref {
                interface ethernet-1/1
                subinterface 1
            }
            input {
                acl-filter ip_tcp type ipv4 {
                }
            }
        }
    }
```

```
--{ * candidate shared default }--[  ]--
# info system mirroring
    system {
        mirroring {
            mirroring-instance 1 {
                admin-state enable
                mirror-source {
                    acl {
                        acl-filter ip_tcp type ipv4 {
                            entry 1000 {
                            }
                        }
                    }
                }
            }
        }
    }
```

3.3.2 Configuring mirroring destinations

Procedure

In a mirroring-instance, you specify the destination for the mirrored traffic. The mirroring destination can be a local destination residing on the same SR Linux node as the mirroring source, or a remote destination where the mirrored traffic is sent via a tunnel. The tunneled traffic can be encapsulated with GRE protocol type 0x6558 or 0x88BE (7250 IXR-6e/10e and 7250 IXR-X3b platforms only).

Example: Local destination

The following example enables a subinterface to be a local mirror destination:

```
--{ * candidate shared default }--[  ]--
# info from running interface ethernet-1/4 subinterface 1
    interface ethernet-1/4 {
        subinterface 1 {
            type local-mirror-dest
            admin-state enable
            vlan {
                encap {
                    single-tagged {
                        vlan-id 1127
                    }
```

```
                }
            }
            local-mirror-destination {
                admin-state enable
            }
        }
    }
```

The following example configures a mirroring-instance where traffic from the mirror source is mirrored to the subinterface enabled as a local mirror destination:

```
--{ * candidate shared default }--[  ]--
# info system mirroring
    system {
        mirroring {
            mirroring-instance 1 {
                admin-state enable
                mirror-source {
                    interface ethernet-2/1 {
                        direction ingress-egress
                    }
                }
                mirror-destination {
                    local ethernet-1/4.1
                }
            }
        }
    }
```

Example: Remote destination using underlay

The following example configures a mirroring-instance that specifies the mirrored traffic be encapsulated into a tunnel within a network-instance. The mirrored traffic is encapsulated into a tunnel using L2oGRE to the remote destination.

```
--{ * candidate shared default }--[  ]--
# info system mirroring
    system {
        mirroring {
            mirroring-instance 1 {
                admin-state enable
                mirror-source {
                    interface ethernet-2/1 {
                        direction ingress-egress
                    }
                }
                mirror-destination {
                    remote {
                        encap l2ogre
                        network-instance IPVRF-1 {
                            tunnel-end-points {
                                src-ipv4 192.168.1.53
                                dst-ipv4 192.168.1.153
                            }
                        }
                    }
                }
            }
        }
    }
```

3.4 Displaying mirroring information

Procedure

Use the **info from state** command to display mirroring configuration information.

Example

```
--{ * candidate shared default }--[  ]--
# info from state system mirroring mirroring-instance 2
    system {
        mirroring {
            mirroring-instance 2 {
                admin-state enable
                oper-state down
                oper-down-reason local-mirror-subif-down
                mirror-source {
                    interface lag1 {
                        direction ingress-egress
                    }
                }
                mirror-destination {
                    local lag25.1
                }
            }
        }
    }
```

3.5 Displaying mirroring statistics

Procedure

On 7220 IXR-D2/D3 platforms, you can display the statistics per mirror destination interface using the **info from state interface * statistics** command. See **out-mirrored-packets** and the **out-mirrored-octets** fields.

On 7220 IXR-D4/D5, 7250 IXR-6e/10e, and 7250 IXR-X3b platforms, mirror destination statistics are not supported per-interface; it is only possible to display per-mirror-destination statistics. The statistics show the number of packets sent to the mirror destination.

On 7220 IXR-D4/D5 platforms, the statistics only include the number of packets mirrored in either the ingress or the egress direction. On 7250 IXR-6e/10e and 7250 IXR-X3b platforms, the statistics include the number of packets in the ingress direction and the number of octets mirrored in either the ingress or the egress direction.

The octet count for ERSPAN includes the GRE header (not just the actual mirror packet). The interfaces that egress the mirrored packet must adjust the MTU size to accommodate that additional GRE header. If the MTU size is smaller than the GRE packet, the mirrored packet is dropped.

There are no packet drop statistics for mirror destinations. The statistics represent all packets that have been successfully mirrored and sent to the mirror destination. It is possible for mirrored packets to be dropped because of over-congestion of multiple mirror sources to the same mirror destination. Mirrored packet drops can also occur because a mirror destination interface can be used for regular data traffic forwarding.

Example: Mirroring statistics on 7220 IXR-D2/D3 platforms

```
--{ running }--[  ]--
# info from state interface ethernet-1/48 statistics | filter fields out-mirror-octets
 out-mirror-packets
    interface ethernet-1/48 {
        statistics {
            out-mirror-octets 0
            out-mirror-packets 0
        }
```

Example: Mirroring statistics on 7220 IXR-D5 platform

```
--{ running }--[  ]--
# info from state system mirroring mirroring-instance * mirror-destination statistics
    system {
        mirroring {
            mirroring-instance eight {
                mirror-destination {
                    statistics {
                        ingress-mirrored-packets 22135
                        egress-mirrored-packets 22132
                    }
                }
            }
            mirroring-instance five {
                mirror-destination {
                    statistics {
                        ingress-mirrored-packets 6353567
                        egress-mirrored-packets 0
                    }
                }
            }
        }
    }
```

Example: Mirroring statistics on 7250 IXR-6e/10e platforms

```
--{ running }--[  ]--
# info from state system mirroring mirroring-instance ixia_one mirror-destination
 statistics
    system {
        mirroring {
            mirroring-instance ixia_one {
                mirror-destination {
                    statistics {
                        ingress-mirrored-packets 7417657
                        ingress-mirrored-octets 10384702600
                        egress-mirrored-octets 0
                    }
                }
            }
        }
    }
```

4 OAM fault and performance tools and protocols

This chapter provides information about the operation, administration, and maintenance (OAM) tools and protocols. The tools and protocols are used for:

- fault detection and isolation
- performance measurement

This chapter covers the following sections:

- MPLS OAM tools and protocols
- IP OAM tools and protocols
- Bidirectional Forwarding Detection

4.1 IP OAM tools and protocols

This section provides information about the IP OAM tools and protocols.

4.1.1 ICMP ping and trace

Overview

Internet Control Message Protocol (ICMP) is part of the IP suite as defined in RFC 792, Internet Control Message Protocol, for IPv4 and RFC 4443, Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. ICMP and ICMPv6 send and receive control and error messages used to manage the behavior of the TCP/IP stack. ICMP and ICMPv6 provide the following:

- debugging tools and error reporting mechanisms to assist in troubleshooting an IP network
- the ability to send and receive error and control messages to far-end IP entities

Ping

The **ping** command uses an echo request message to elicit an echo response from a host or gateway. The **ping6** command is the IPv6 version of the **ping** command. See Performing an ICMP ping for more information.

Traceroute

The traceroute command is used to trace the route that the packets take from the current system to the destination. It uses the time to live (TTL) parameter to elicit an ICMP time exceeded response from each gateway along the path to the host. The **traceroute6** command is the IPv6 version of the **traceroute** command. See Performing an ICMP trace for more information.

4.1.1.1 Performing an ICMP ping

Procedure

Use the **ping** (IPv4) or **ping6** (IPv6) command to contact an IP address. Use this command in any mode.

Example: ping for IPV4

```
--{ running }--[  ]--
# ping 192.168.1.1 network-instance default
Pinging 192.168.1.1 in srbase-default
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.027 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.032 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.030 ms
^C
--- 192.168.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 6165ms
rtt min/avg/max/mdev = 0.027/0.030/0.033/0.005 ms
```

4.1.1.2 Performing an ICMP trace

Procedure

To display the path a packet takes to a destination, use the **traceroute** (IPv4) or **traceroute6** (IPv6) command.

To trace the route using TCP SYN packets instead of UDP or ICMP echo packets, use the **tcptraceroute** command.

Example: traceroute for IPv4

```
--{ running }--[  ]--
# traceroute 1.1.1.1  network-instance mgmt
Using network instance srbase-mgmt
traceroute to 10.1.1.1 (10.1.1.1), 30 hops max, 60 byte packets
 1  172.18.18.1 (172.18.18.1)  1.268 ms  1.260 ms  1.256 ms
 2  172.21.40.1 (172.21.40.1)  1.253 ms  1.848 ms  1.851 ms
 3  172.22.35.230 (172.22.35.230)  1.835 ms  1.834 ms  1.828 ms
 4  66.201.62.1 (66.201.62.1)  3.222 ms  3.222 ms  3.216 ms
 5  66.201.34.17 (66.201.34.17)  5.474 ms  5.475 ms  5.480 ms
 6  * * *
 7  206.81.81.10 (206.81.81.10)  32.577 ms  32.542 ms  32.400 ms
 8  10.1.1.1 (10.1.1.1)  22.627 ms  22.637 ms  22.638 ms
```

4.1.2 TWAMP

Two-Way Active Measurement Protocol (TWAMP) is a standards-based method to measure the IP performance between two devices including packet loss, delay, and jitter. TWAMP leverages the methodology and architecture of One-Way Active Measurement Protocol (OWAMP) to define a method to measure two-way or round-trip metrics.

Components

The following are the four logical entities in TWAMP:

- control client: initiates the TWAMP control session and negotiates the session information to be used and the tests to be performed with the server
- server: negotiates with the control client request to establish the control session
- session sender: transmits test packets to the session reflector
- session reflector: transmits a packet to the session sender in response to each packet it receives

The control client and session sender are implemented in one physical device which is referred to as the client. The server and session reflector are implemented in a second physical device which is referred to as the server. The router acts as the server and the session reflector.

See Configuring a TWAMP server for more information about steps to configure a TWAMP server.

Protocols

The following protocols are used in TWAMP sessions:

- TWAMP control protocol: used to establish and manage control sessions between the control client and the server
- TWAMP test protocol: used to generate and send test traffic between the session sender and session reflector, and to measure network performance metrics like delay

Establishing a control session

The control client initiates a TCP connection and exchanges TWAMP control messages over this connection. The server accepts the TCP control session from the control client and responds with a server greeting message. This greeting includes the modes that are supported by the server. The modes are in the form of a bit mask. Each bit in the mask represents a functionality supported on the server.

Server mode support includes:

- unauthenticated server
- individual session control (mode bit 4: value 16)
- reflected octets (mode bit 5: value 32)
- symmetrical size test packet (mode bit 6: value 64)

To start testing, the control client communicates the test parameters to the server, requesting any of the modes that the server supports. If the server agrees to conduct the described tests, the test begins as soon as the control client sends a start sessions or start-n-session message.

Executing a test session

The session sender initiates the test session by sending a stream of UDP-based TWAMP test packets to the session reflector. The session reflector responds to each received packet with a UDP-response TWAMP test packet. The exchange of TWAMP test PDUs is referred to as a TWAMP test. The session sender calculates the various delay and loss metric based on the received TWAMP test PDUs . The TWAMP test PDU does not achieve symmetrical packet size in both directions unless the frame is padded with a minimum of 27 bytes. The session sender is responsible for applying the required padding. After the frame is appropriately padded, the session reflector reduces the padding by the number of bytes needed to provide symmetry.

The control client eventually closes the control session, marking the end of the measurement process.

TWAMP statistics

The following TWAMP statistics are available in SR Linux:

- system-level TWAMP statistics
- server statistics
- client connection statistics
- control connection statistics
- session reflector statistics

See Displaying TWAMP statistics for more information.

A clear command is available at the server network-instance level to clear all test session transmit and receive statistics and error counters. See Clearing TWAMP session statistics for more information.

4.1.2.1 Configuring a TWAMP server

Procedure

To configure a TWAMP server (server and session reflector) for a network instance, use the **oam twamp** command and specify the network instance, client (control client and session sender) parameters, and server parameters as shown in the example.

Example: Configure TWAMP server

The following example configures a TWAMP server.

```
--{ + candidate shared default }--[  ]--
# info oam twamp
    oam {
        twamp {
            server {
                network-instance default {
                    admin-state enable
                    servwait 900
                    control-packet-dscp 12
                    enforce-test-session-start-time true
                    client-connection 192.15.20.9/32 {
                        maximum-connections 32
                        maximum-sessions 32
                    }
                }
            }
        }
    }
```

4.1.2.2 Displaying TWAMP statistics

Procedure

To display system-level TWAMP statistics, use the **info from state oam twamp** command.

Example: Displaying TWAMP statistics

The following example displays TWAMP statistics.

```
--{ + candidate shared default }--[  ]--
# info from state oam twamp
    oam {
        twamp {
            server {
                network-instance default {
                    admin-state enable
                    oper-state up
                    servwait 60
                    control-packet-dscp CS7
  //this will be removed form yang//
                    enforce-test-session-start-time true
                    maximum-connections 64
                    maximum-sessions 128
                    modes [
                        unauthenticated
                        individual-session-control
                        reflect-octets
                        symmetrical-size
                    ]
                    statistics {
                        test-sessions-active 1
                        test-sessions-completed 0
                        test-sessions-rejected 0
                        test-sessions-aborted 0
                        test-packets-received 2
                        test-packets-transmitted 2
                        control-connections-active 1
                        control-connections-rejected 0
                    }
                    client-connection 10.32.5.0/24 {
                        maximum-connections 64
                        maximum-sessions 128
                        statistics {
                            test-sessions-active 1
                            test-sessions-completed 0
                            test-sessions-rejected 0
                            test-sessions-aborted 0
                            test-packets-received 2
                            test-packets-transmitted 2
                            control-connections-active 1
                            control-connections-rejected 0
                        }
                    }
                    client-connection 10.11.1.0/24 {
                        maximum-connections 64
                        maximum-sessions 128
                        statistics {
                            test-sessions-active 0
                            test-sessions-completed 0
                            test-sessions-rejected 0
                            test-sessions-aborted 0
                            test-packets-received 0
                            test-packets-transmitted 0
                            control-connections-active 0
                            control-connections-rejected 0
                        }
                    }
                    client-connection 10.12.1.0/24 {
                        maximum-connections 32
```

```
                        maximum-sessions 32
                        statistics {
                            test-sessions-active 0
                            test-sessions-completed 0
                            test-sessions-rejected 0
                            test-sessions-aborted 0
                            test-packets-received 0
                            test-packets-transmitted 0
                            control-connections-active 0
                            control-connections-rejected 0
                        }
                    }
                    client-connection 2001:db8:101:1:1::/120 {
                        maximum-connections 64
                        maximum-sessions 128
                        statistics {
                            test-sessions-active 0
                            test-sessions-completed 0
                            test-sessions-rejected 0
                            test-sessions-aborted 0
                            test-packets-received 0
                            test-packets-transmitted 0
                            control-connections-active 0
                            control-connections-rejected 0
                        }
                    }
                    client-connection 2001:db8:101:1:1::/120 {
                        maximum-connections 32
                        maximum-sessions 32
                        statistics {
                            test-sessions-active 0
                            test-sessions-completed 0
                            test-sessions-rejected 0
                            test-sessions-aborted 0
                            test-packets-received 0
                            test-packets-transmitted 0
                            control-connections-active 0
                            control-connections-rejected 0
                        }
                    }
                    client-connection 2001:db8:101:1:1::/120 {
                        maximum-connections 64
                        maximum-sessions 128
                        statistics {
                            test-sessions-active 0
                            test-sessions-completed 0
                            test-sessions-rejected 0
                            test-sessions-aborted 0
                            test-packets-received 0
                            test-packets-transmitted 0
                            control-connections-active 0
                            control-connections-rejected 0
                        }
                    }
                    control-connection 10.32.5.0 client-tcp-port 58116 server-ip 10.20.1.3 server-tcp-
port 862 {
                        state active
                        control-packet-dscp 20
                        statistics {
                            test-sessions-active 1
                            test-sessions-completed 0
                            test-sessions-rejected 0
                            test-sessions-aborted 0
                            test-packets-received 2
```

```
                            test-packets-transmitted 2
                        }
                    }
                    session-reflector {
                        test-session 10.32.5.0 sender-udp-port 20100 reflector-ip 10.20.1.3 reflector-
udp-port 862 {
                            test-session-id 0A:14:01:03:EA:0B:06:CC:1A:36:F3:B2:A3:97:A2:55
                            parent-connection-client-ip 32.32.5.2
                            parent-connection-client-tcp-port 58116
                            parent-connection-server-ip 10.20.1.3
                            parent-connection-server-tcp-port 862
                            test-packet-dscp 0
                            last-sequence-number-transmitted 1
                            last-sequence-number-received 0
                            statistics {
                                test-packets-received 2
                                test-packets-transmitted 2
                            }
                        }
                    }
                }
            }
            statistics {
                dropped-connections {
                    tcp-connection-closed 0
                    tcp-connection-fatal-error 0
                    tcp-unexpected-event 0
                    message-send-error 0
                    memory-allocation-error 0
                    no-client-prefix-match 0
                    maximum-global-limit-exceed 0
                    maximum-prefix-limit-exceed 0
                    unspecified-mode 0
                    unsupported-mode 0
                    control-command-not-valid 0
                    incorrect-stop-session-count 0
                    connection-timeout 0
                    no-internal-resource 0
                    non-zero-sid-in-client-control-message 0
                    invalid-invalid-hmac 0
                }
                dropped-connection-states {
                    idle 0
                    setup-wait 0
                    started 0
                    active 0
                    process-started 0
                    process-stop 0
                    process-tw-session 0
                }
                rejected-session {
                    invalid-ip-address-version 0
                    non-local-ip-destination 0
                    bad-type-p 0
                    padding-too-big 0
                    non-zero-mbz-value 0
                    non-zero-session-sender-sid 0
                    timeout-too-large 0
                    maximum-global-session-exceed 0
                    maximum-prefix-session-exceed 0
                    client-source-ip-unreachable 0
                    udp-port-in-use 0
                    duplicate-session 0
                    no-internal-resource 0
```

```
                refwait-timeout 0
            }
            dropped-test-packet {
                incorrect-packet-size 0
                incorrect-source-address 0
                arrived-before-start-time 0
                no-start-sessions 0
                invalid-error-estimate 0
                reply-error 0
                invalid-server-octets 0
                invalid-symmetric-mbz 0
            }
        }
    }
}
```

4.1.2.3 Clearing TWAMP session statistics

Procedure

You can clear the TWAMP session statistics for each network instance using the **tools oam twamp server network-instance default clear** command.

Example: Clearing TWAMP session statistics

The following example clears TWAMP session statistics for each network instance.

```
--{ + candidate shared default }--[  ]--
# tools oam twamp server network-instance default clear
```

4.1.3 STAMP

The Simple Two-Way Active Measurement Protocol (STAMP) defined in RFC 8762 is a standards-based method to measure the IP performance without the use of a control channel to pre-signal session parameters.

The PDU structure allows for the collection of frame delay, frame delay range, inter-frame delay variation, and frame loss ratio. The RFC 8972 STAMP Optional Extensions specification maintains the existing structure of the STAMP PDU but redefines existing fields and adds the capability to include TLVs. SR Linux supports RFC 8762 and the structural changes with TLV processing in the options draft RFC 8972.

STAMP operation

For each routed network instance, the STAMP session sender transmits STAMP test packets to the destination UDP port of the session reflector. The session reflector receives the packets, processes the STAMP test packet, and sends them back to the session sender. The session sender receives the reflected packets and uses the timestamps and sequence numbers to calculate delay and loss performance metrics. The session reflector supports a prefix list which filters based on IPv4 or IPv6 addressing. The reflector is stateful and uses the tuple sSIP, DIP, SP, DP, or SSID to identify individual STAMP test sessions.

See Configuring STAMP session reflector for more information about how to configure a session reflector.

Session sender packet format

STAMP defines the STAMP test request packets sent by the STAMP session sender to the STAMP session reflector as probes for performance measurement. The following table lists the key protocol elements.

Table 3: Fields in a test request packet

Field	Description
Sequence Number	Packet sequence number generated based on the transmission sequence. For each new session, its value starts at 0 and is incremented by one with each transmitted packet.
Timestamp	Timestamp when a test packet is sent.
Error Estimate	Estimated error field. The format is as follows: • S bit is set to 0 regardless of time synchronization. • Z bit is set to 0 because the timestamp format is NTP. • Scale bits are set to 0. • Multiplier bits are non-zero.
SSID	Session Sender ID (SSID) automatically generated by the system.
MBZ	Must-Be-Zero (MBZ). The value must be 0. This field is used to ensure data packet symmetry between the session sender and session reflector.

Session reflector packet format

STAMP defines the STAMP test response packets reflected by the STAMP session reflector to the STAMP session sender. The following table lists the key protocol elements.

Table 4: Fields in a test response packet

Fields	Description
Sequence Number	Packet sequence number generated based on the transmission sequence. For each new session, its value starts at 0 and is incremented by one with each transmitted packet.
Timestamp	Timestamp when a test packet is transmitted from the session reflector
Error Estimate	Estimated error field. The format is as follows: • S bit is set to 0 regardless of time synchronization. • Z bit is set to 0 because the timestamp format is NTP. • Scale bits are set to 0. • Multiplier bits are non-zero.
SSID	Session Sender ID (SSID) automatically generated by the system.
MBZ	Must-Be-Zero. The value must be 0. This field is used to ensure data packet symmetry between the Session-Sender and Session-Reflector.

Fields	Description
Receive Timestamp	Timestamp when a test packet is received on the session reflector.
Session-Sender Sequence Number	It is copied from the Sequence Number field of the STAMP Test request packet.
Session-Sender Timestamp	It is copied from the Timestamp field of the STAMP Test request packet.
Session-Sender Error Estimate	It is copied from the Error Estimate field of the STAMP Test request packet.
Session-Sender TTL	TTL value of a packet.

Interoperability of STAMP and TWAMP Light

The following guidelines ensure that interoperability exists between STAMP and TWAMP Light by defining rules for packet processing based on packet size and content, particularly the 45th byte, to distinguish between the two protocols:

- UDP packets with a length less than 44 bytes are processed using TWAMP Light processing rules, which involves simple padding and symmetrical packet size handling.
- UDP packets with a length equal to 44 bytes are processed as STAMP packets.
- For UDP packets with a length equal to 45 bytes or more, the 45th byte is checked for the flags structure (100xxxxx).
  - If found, the packets are processed as STAMP packets.
  - If not found, the packet is assumed to be a TWAMP Light padded packet and processed accordingly. The TWAMP Light packet uses all zeros padding to avoid matching the 100xxxxx pattern by accident.
- Multiple TLVs in a STAMP test packet are parsed using the length field.
- If a TWAMP Light test packet mistakenly matches the 100xxxxx pattern at byte 45, the reflector attempts to parse the TLV. Failure to parse results in marking the byte as 110xxxxx and halting further STAMP TLV processing. However, the base STAMP packet continues to be processed.
- TWAMP Light packets arriving on a STAMP session reflector must use all zeros padding to avoid unintentional mismatching.

STAMP statistics

The following STAMP statistics are available in SR Linux:

- system-level session reflector statistics
- session reflector statistics for each network instance
- test session statistics

See Displaying STAMP statistics for more information.

4.1.3.1 Configuring STAMP session reflector

Procedure

To configure a STAMP session reflector for a network instance, use the **oam stamp** command and specify the network instance, IP address prefix, and the UDP port as shown in the example.

Example: Configuring STAMP session reflector

The following example configures a session reflector.

```
--{ + candidate shared default }--[  ]--
# info oam stamp
    oam {
        stamp {
            session-reflector {
                network-instance default {
                    description test
                    admin-state enable
                    udp-port 862
                    ip-prefix 192.20.13.20/32 {
                    }
                }
            }
        }
    }
```

4.1.3.2 Displaying STAMP statistics

Procedure

To display system-level STAMP session reflector statistics, use the **info from state oam stamp** command.

Example: Displaying STAMP statistics

The following example displays STAMP statistics.

```
--{ + candidate shared default }--[  ]--
# info from state oam stamp
  oam {
        stamp {
            session-reflector {
                inactivity-timer 900
                statistics {
                    test-frames-received 400
                    test-frames-sent 400
                    test-session-count 4
                    reflector-table-entries-full 0
                    packet-discards-on-reception 0
                    packet-discards-on-transmission 0
                    session-reflector-not-found 0
                    reflectors-configured 1
                    reflectors-operational 1
                    reflectors-not-operational 0
                }
                network-instance default {
                    admin-state enable
                    udp-port 862
```

```
                    oper-state up
                    ip-prefix  10.11.1.0/24 {
                    }
                    ip-prefix 10.10.11.0/24 {
                    }
                    ip-prefix 10.10.12.0/24 {
                    }
                    ip-prefix 10.10.14.0/24 {
                    }
                    ip-prefix 10.20.1.0/24 {
                    }
                    ip-prefix 10.12.1.0/24 {
                    }
                    ip-prefix 10.13.1.0/24 {
                    }
                    ip-prefix 10.14.1.0/24 {
                    }
                    ip-prefix 2001:db8:101:1:1/120 {
                    }
                    ip-prefix 2001:db8:102:1:1/120 {
                    }
                    ip-prefix 2001:db8:103:1:1/120 {
                    }
                    ip-prefix 2001:db8:104:1:1/120 {
                    }
                    ip-prefix 2001:db8:105:1:1/120 {
                    }
                    ip-prefix 2001:db8:106:1:1/120 {
                    }
                    ip-prefix 2001:db8:107:1:1/120 {
                    }
                    ip-prefix 2001:db8:108:1:1/120 {
                    }
                    statistics {
                        test-frames-received 400
                        test-frames-sent 400
                        test-sessions 4
                        prefix-match-failure 0
                        session-reflector-udp-port-registration-failure 0
                        malformed-packet 0
                        packet-discards-source-destination-equal 0
                    }
                    test-session-statistics 10.20.1.3 session-sender-udp 44000 session-reflector-ip
 11.20.1.2 session-reflector-udp 862 session-identifier 1736 {
                        last-sequence-number-received 99
                        last-sequence-number-transmitted 99
                        test-frames-received 100
                        test-frames-sent 100
                        malformed-tlv 0
                    }
                    test-session-statistics 10.10.3.3 session-sender-udp 44000 session-reflector-ip
 20.10.3.2 session-reflector-udp 862 session-identifier 1737 {
                        last-sequence-number-received 99
                        last-sequence-number-transmitted 99
                        test-frames-received 100
                        test-frames-sent 100
                        malformed-tlv 0
                    }
                    test-session-statistics 2001:db8:103:1:1 session-sender-udp 44000 session-
reflector-ip fc00::b14:102 session-reflector-udp 862 session-identifier 1738 {
                        last-sequence-number-received 99
                        last-sequence-number-transmitted 99
                        test-frames-received 100
                        test-frames-sent 100
```

```
                        malformed-tlv 0
                    }
                    test-session-statistics 2001:db8:104:1:1 session-sender-udp 44000 session-
reflector-ip fc00::140a:302 session-reflector-udp 862 session-identifier 1739 {
                        last-sequence-number-received 99
                        last-sequence-number-transmitted 99
                        test-frames-received 100
                        test-frames-sent 100
                        malformed-tlv 0
                    }
                }
```

4.2 MPLS OAM tools and protocols

This section provides information about the MPLS OAM tools and protocols.

4.2.1 LSP ping and trace

The LSP diagnostics include implementations of LSP ping and LSP trace based on RFC 8029, *Detecting Multiprotocol Label Switched (MPLS) Data Plane Failures*. LSP ping provides a mechanism to detect data plane failures in MPLS LSPs. LSP ping and LSP trace are modeled after the ICMP echo request or reply used by ping and trace to detect and localize faults in IP networks.

For a specific LDP FEC, LSP ping verifies whether the packet reaches the egress label edge router (LER), while for LSP trace, the packet is sent to the control plane of each transit Label Switching Router (LSR) that performs various checks to see if it is intended to be a transit LSR for the path.

The downstream mapping TLV is used in LSP ping and LSP trace to provide a mechanism for the sender and responder nodes to exchange and validate interface and label stack information for each downstream hop in the path of an LDP FEC.

See the following topics for more information about performing LSP ping and trace:

- Performing an LSP ping to an LDP tunnel endpoint
- Performing an LSP trace for an LDP tunnel
- Performing an LSP ping to a segment routing prefix
- Performing an LSP trace to a segment routing prefix
- Performing an LSP ping to an uncolored SR-MPLS TE policy
- Performing an LSP trace to an uncolored SR-MPLS TE policy

4.2.1.1 ECMP considerations for LSP ping and LSP trace

If an LSP trace is initiated without the destination IP address, the sender node does not include multi-path information in the Downstream Mapping TLV of the echo request message (multipath type=0). The responder node replies with a Downstream Mapping TLV for each outgoing interface which is part of the ECMP next hop set for the FEC. The sender node selects the first Downstream Mapping TLV to use for subsequent probes one hop further toward the destination.

If an LSP trace is initiated with the destination IP address, the sender node includes the multipath information in the Downstream Mapping TLV in the echo request message (multipath type=8). The **ecmp-interface-select** and **ecmp-next-hop-select** options allow the LER to exercise a specific ECMP path. If both the options are specified, the **ecmp-interface-select** takes precedence. The **ecmp-interface-select** and **ecmp-next-hop-select** options can be used to direct the echo request message at the sender node to be sent out to a specific outgoing interface which is part of an ECMP path set for the FEC.

4.2.1.2 LSP ping and trace for LDP tunnels

To check connectivity and trace the path to any midpoint or endpoint of an LDP tunnel, SR Linux supports the following OAM commands:

- **tools oam lsp-ping ldp fec** *<prefix>*
- **tools oam lsp-trace ldp fec** *<prefix>*

Supported parameters include **destination-ip**, **source-ip**, **timeout**, **ecmp-next-hop-select**, and **traffic-class**. However, the only mandatory parameter is **fec**.

Results from the lsp-ping and lsp-trace operations are displayed using **info from state** commands.

4.2.1.2.1 Performing an LSP ping to an LDP tunnel endpoint

Procedure

To check connectivity to an LDP tunnel endpoint, use the **tools oam lsp-ping ldp** command, specifying the IPv4 and IPv6 FEC prefix of the LDP tunnel. To display the results, use the **info from state oam lsp-ping ldp** command, specifying the session ID output from the **lsp-ping**.

Example: Perform an LSP ping to an LDP tunnel endpoint (IPv4)

```
--{ + running }--[  ]--
# tools oam lsp-ping ldp fec 10.20.1.6/32
/oam/lsp-ping/ldp/fec[prefix=10.20.1.6/32]:
    Initiated LSP Ping to prefix 10.20.1.6/32 with session id 49152
```

Example: Display results of the LSP ping (IPv4)

```
--{ + running }--[  ]--
# info from state oam lsp-ping ldp fec 10.20.1.6/32 session-id 49152
    oam {
        lsp-ping {
            ldp {
                fec 10.20.1.6/32 {
                    session-id 49152 {
                        test-active false
                        statistics {
                            round-trip-time {
                                minimum 4292
                                maximum 4292
                                average 4292
                                standard-deviation 0
                            }
                        }
                        path-destination {
                            ip-address 127.0.0.1
                        }
```

```
                    sequence 1 {
                        probe-size 48
                        request-sent true
                        out-interface ethernet-1/33.1
                        reply {
                            received true
                            reply-sender 10.20.1.6
                            udp-data-length 40
                            mpls-ttl 255
                            round-trip-time 4292
                            return-code replying-router-is-egress-for-fec-at-stack-depth-n
                            return-subcode 1
                        }
                    }
                }
            }
        }
    }
}
```

Example: Perform an LSP ping to an LDP tunnel endpoint (IPv6)

```
--{ + running }--[  ]--
# tools oam lsp-ping ldp fec fc00::a14:106/128
/oam/lsp-ping/ldp/fec[prefix=fc00::a14:106/128]:
    Initiated LSP Ping to prefix fc00::a14:106/128 with session id 49169
```

Example: Display results of the LSP ping (IPv6)

```
--{ + running }--[  ]--
# info from state oam lsp-ping ldp fec fc00::a14:106/128 session-id 49169
    oam {
        lsp-ping {
            ldp {
                fec fc00::a14:106/128 {
                    session-id 49169 {
                        test-active false
                        statistics {
                            round-trip-time {
                                minimum 47539
                                maximum 47539
                                average 47539
                                standard-deviation 0
                            }
                        }
                        path-destination {
                            ip-address ::ffff:127.0.0.0
                        }
                        sequence 1 {
                            probe-size 60
                            request-sent true
                            out-interface ethernet-1/31.1
                            reply {
                                received true
                                reply-sender fc00::a14:106
                                udp-data-length 40
                                mpls-ttl 255
                                round-trip-time 47539
                                return-code replying-router-is-egress-for-fec-at-stack-
depth-n
                                return-subcode 1
                            }
```

```
                        }
                    }
                }
            }
        }
    }
```

4.2.1.1.2.2 Performing an LSP trace for an LDP tunnel

Procedure

To trace the path to any midpoint or endpoint of an LDP tunnel, use the **tools oam lsp-trace** command, specifying the IPv4 and IPv6 FEC prefix of the LDP tunnel. To display the results, use the **info from state oam lsp-trace ldp** command, specifying the session ID output from the **lsp-trace**.

Example: Perform an LSP trace to an LDP tunnel endpoint (IPv4)

```
--{ + running }--[  ]--
# tools oam lsp-trace ldp fec 10.20.1.6/32
/oam/lsp-trace/ldp/fec[prefix=10.20.1.6/32]:
    Initiated LSP Trace to prefix 10.20.1.6/32 with session id 49153
```

Example: Display results of the LSP trace (IPv4)

```
--{ + running }--[  ]--
# info from state oam lsp-trace ldp fec 10.20.1.6/32 session-id 49153
    oam {
        lsp-trace {
            ldp {
                fec 10.20.1.6/32 {
                    session-id 49153 {
                        test-active false
                        path-destination {
                            ip-address 127.0.0.1
                        }
                        hop 1 {
                            probe 1 {
                                probe-size 76
                                probes-sent 1
                                reply {
                                    received true
                                    reply-sender 10.20.1.2
                                    udp-data-length 60
                                    mpls-ttl 1
                                    round-trip-time 4824
                                    return-code label-switched-at-stack-depth-n
                                    return-subcode 1
                                }
                                downstream-detailed-mapping 1 {
                                    mtu 1500
                                    address-type ipv4-numbered
                                    downstream-router-address 10.10.4.4
                                    downstream-interface-address 10.10.4.4
                                    mpls-label 1 {
                                        label 2002
                                        protocol ldp
                                    }
                                }
                            }
                        }
```

```
                        hop 2 {
                            probe 1 {
                                probe-size 76
                                probes-sent 1
                                reply {
                                    received true
                                    reply-sender 10.20.1.4
                                    udp-data-length 60
                                    mpls-ttl 2
                                    round-trip-time 4693
                                    return-code label-switched-at-stack-depth-n
                                    return-subcode 1
                                }
                                downstream-detailed-mapping 1 {
                                    mtu 1500
                                    address-type ipv4-numbered
                                    downstream-router-address 10.10.9.6
                                    downstream-interface-address 10.10.9.6
                                    mpls-label 1 {
                                        label 2000
                                        protocol ldp
                                    }
                                }
                            }
                        }
                        hop 3 {
                            probe 1 {
                                probe-size 76
                                probes-sent 1
                                reply {
                                    received true
                                    reply-sender 10.20.1.6
                                    udp-data-length 32
                                    mpls-ttl 3
                                    round-trip-time 4597
                                    return-code replying-router-is-egress-for-fec-at-
stack-depth-n
                                    return-subcode 1
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

Example: Perform an LSP trace to an LDP tunnel endpoint (IPv6)

```
--{ + running }--[  ]--
# tools oam lsp-trace ldp fec fc00::a14:106/128
/oam/lsp-trace/ldp/fec[prefix=fc00::a14:106/128]:
    Initiated LSP Trace to prefix fc00::a14:106/128 with session id 49168
```

Example: Display results of the LSP trace (IPv6)

```
--{ + running }--[  ]--
# info from state oam lsp-trace ldp fec fc00::a14:106/128 session-id 49168
    oam {
        lsp-trace {
            ldp {
                fec fc00::a14:106/128 {
```

```
                    session-id 49168 {
                        test-active false
                        path-destination {
                            ip-address ::ffff:127.0.0.0
                        }
                        hop 1 {
                            probe 1 {
                                probe-size 112
                                probes-sent 1
                                reply {
                                    received true
                                    reply-sender fc00::a14:102
                                    udp-data-length 84
                                    mpls-ttl 1
                                    round-trip-time 41527
                                    return-code label-switched-at-stack-depth-n
                                    return-subcode 1
                                }
                                downstream-detailed-mapping 1 {
                                    mtu 1500
                                    address-type ipv6-numbered
                                    downstream-router-address fe80::201:4ff:feff:1e
                                    downstream-interface-address fe80::201:4ff:feff:1e
                                    mpls-label 1 {
                                        label 2008
                                        protocol ldp
                                    }
                                }
                            }
                        }
                        hop 2 {
                            probe 1 {
                                probe-size 112
                                probes-sent 1
                                reply {
                                    received true
                                    reply-sender fc00::a14:104
                                    udp-data-length 84
                                    mpls-ttl 2
                                    round-trip-time 76569
                                    return-code label-switched-at-stack-depth-n
                                    return-subcode 1
                                }
                                downstream-detailed-mapping 1 {
                                    mtu 1500
                                    address-type ipv6-numbered
                                    downstream-router-address fe80::201:6ff:feff:3
                                    downstream-interface-address fe80::201:6ff:feff:3
                                    mpls-label 1 {
                                        label 2001
                                        protocol ldp
                                    }
                                }
                            }
                        }
                        hop 3 {
                            probe 1 {
                                probe-size 112
                                probes-sent 1
                                reply {
                                    received true
                                    reply-sender fc00::a14:106
                                    udp-data-length 32
                                    mpls-ttl 3
```

```
                                    round-trip-time 41739
                                    return-code replying-router-is-egress-for-fec-at-
stack-depth-n
                                    return-subcode 1
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

4.2.1.3 LSP ping and trace for segment routing tunnels

To check connectivity and trace the path to any midpoint or endpoint of an SR-ISIS shortest path tunnel, SR Linux supports the following OAM commands:

- **tools oam lsp-ping sr-isis prefix-sid** <*prefix*>
- **tools oam lsp-trace sr-isis prefix-sid** <*prefix*>

Supported parameters include **destination-ip**, **source-ip**, **timeout**, **ecmp-next-hop-select**, **igp-instance**, and **traffic-class**. However, the only mandatory parameter is the **prefix-sid**.

Results from the lsp-ping and lsp-trace operations are displayed using **info from state** commands.

In the case of ECMP, even when the destination IP is configured, the SR Linux node may not exercise all NHLFEs.

4.2.1.3.1 Performing an LSP ping to a segment routing prefix

Procedure

To check connectivity to a segment routing prefix, use the **tools oam lsp-ping sr-isis** command. To display the results, use the **info from state oam lsp-ping sr-isis** command, specifying the session ID output from the **lsp-ping**.

Example: Perform an LSP ping to a destination segment routing prefix

```
# tools oam lsp-ping sr-isis prefix-sid 10.20.1.6/32
/oam/lsp-ping/sr-isis/prefix-sid[prefix=10.20.1.6/32]:
    Initiated LSP Ping to prefix 10.20.1.6/32 with session id 49152
```

Example: Display results of the LSP ping

```
--{ + running }--[  ]--
# info from state oam lsp-ping sr-isis prefix-sid 10.20.1.6/32 session-id 49152
    oam {
        lsp-ping {
            sr-isis {
                prefix-sid 10.20.1.6/32 {
                    session-id 49152 {
                        test-active false
                        statistics {
                            round-trip-time {
                                minimum 4292
                                maximum 4292
```

```
                                average 4292
                                standard-deviation 0
                            }
                        }
                        path-destination {
                            ip-address 127.0.0.1
                        }
                        sequence 1 {
                            probe-size 48
                            request-sent true
                            out-interface ethernet-1/33.1
                            reply {
                                received true
                                reply-sender 10.20.1.6
                                udp-data-length 40
                                mpls-ttl 255
                                round-trip-time 4292
                                return-code replying-router-is-egress-for-fec-at-stack-depth-n
                                return-subcode 1
                            }
                        }
                    }
                }
            }
        }
    }
```

4.2.1.3.2 Performing an LSP trace to a segment routing prefix

Procedure

To trace the path to any midpoint or endpoint of a segment routing tunnel, use the **tools oam lsp-trace** command. To display the results, use the **info from state oam lsp-trace sr-isis** command, specifying the session ID output from the **lsp-trace**.

Example: Perform an LSP trace to a destination segment routing prefix

```
# tools oam lsp-trace sr-isis prefix-sid 10.20.1.6/32
/oam/lsp-trace/sr-isis/prefix-sid[prefix=10.20.1.6/32]:
    Initiated LSP Trace to prefix 10.20.1.6/32 with session id 49153
```

Example: Display results of the LSP trace

```
--{ + running }--[  ]--
# info from state oam lsp-trace sr-isis prefix-sid 10.20.1.6/32 session-id 49153
    oam {
        lsp-trace {
            sr-isis {
                prefix-sid 10.20.1.6/32 {
                    session-id 49153 {
                        test-active false
                        path-destination {
                            ip-address 127.0.0.1
                        }
                        hop 1 {
                            probe 1 {
                                probe-size 76
                                probes-sent 1
                                reply {
                                    received true
```

```
                                reply-sender 10.20.1.2
                                udp-data-length 60
                                mpls-ttl 1
                                round-trip-time 2768
                                return-code label-switched-at-stack-depth-n
                                return-subcode 1
                            }
                            downstream-detailed-mapping 1 {
                                mtu 1500
                                address-type ipv4-numbered
                                downstream-router-address 10.10.4.4
                                downstream-interface-address 10.10.4.4
                                mpls-label 1 {
                                    label 27000
                                    protocol isis
                                }
                            }
                        }
                    }
                    hop 2 {
                        probe 1 {
                            probe-size 76
                            probes-sent 1
                            reply {
                                received true
                                reply-sender 10.20.1.4
                                udp-data-length 60
                                mpls-ttl 2
                                round-trip-time 3414
                                return-code label-switched-at-stack-depth-n
                                return-subcode 1
                            }
                            downstream-detailed-mapping 1 {
                                mtu 1500
                                address-type ipv4-numbered
                                downstream-router-address 10.10.9.6
                                downstream-interface-address 10.10.9.6
                                mpls-label 1 {
                                    label 27000
                                    protocol isis
                                }
                            }
                        }
                    }
                    hop 3 {
                        probe 1 {
                            probe-size 76
                            probes-sent 1
                            reply {
                                received true
                                reply-sender 10.20.1.6
                                udp-data-length 32
                                mpls-ttl 3
                                round-trip-time 4429
                                return-code replying-router-is-egress-for-fec-at-stack-depth-n
                                return-subcode 1
                            }
                        }
                    }
                }
            }
        }
    }
}
```

4.2.1.4 LSP ping and trace for uncolored SR-MPLS TE policy

To check connectivity and trace the path of a segment routing (SR) traffic-engineered (TE) tunnel using an uncolored SR-MPLS TE policy, SR Linux supports the following OAM commands:

- **tools oam lsp-ping te-policy sr-uncolored policy** *<policy name>* **protocol-origin** *<value>*
- **tools oam lsp-trace te-policy sr-uncolored policy** *<policy-name>* **protocol-origin** *protocol-origin <value>*

Supported parameters include **destination-ip**, **source-ip**, **interval**, **segment-list-index**, **timeout**, **ecmp-interface-select mpls-ttl**, **ecmp-next-hop-select**, **send-count**, **traffic-class**, and **probe-size**. The mandatory parameters are, **policy** and **protocol-origin**.

Results from the lsp-ping and lsp-trace operations are displayed using **info from state** commands.

4.2.1.4.1 Performing an LSP ping to an uncolored SR-MPLS TE policy

Procedure

To check connectivity to a SR-TE tunnel using an uncolored SR-MPLS TE policy, execute the **tools oam lsp-ping te-policy sr-uncolored policy** command, specifying the uncolored SR-MPLS TE policy name and the protocol origin. To display the results, use the **info from state oam lsp-ping te-policy sr-uncolored policy protocol-origin session-id** command, specifying the session ID output from the **lsp-ping**.

Example: Perform an LSP ping to an uncolored SR-MPLS TE policy

```
--{ + running }--[  ]--
# tools oam lsp-ping te-policy sr-uncolored policy polABCEF protocol-origin local
/:
Initiated LSP Ping for TE-policy polABCEF with session id 49152.
Please check "info from state oam" for result
```

Example: Display results of the LSP ping

```
--{ + running }--[  ]--
# info from state oam lsp-ping te-policy sr-uncolored policy polABCEF protocol-origin local session-id
 49152
    oam {
        lsp-ping {
            te-policy {
                sr-uncolored {
                    policy polABCEF protocol-origin local {
                        session-id 49152 {
                            test-active false
                            statistics {
                                round-trip-time {
                                    minimum 83
                                    maximum 83
                                    average 83
                                    standard-deviation 0
                                }
                            }
                            path-destination {
                                ip-address 127.0.0.1
                            }
```

```
                            sequence 1 {
                                probe-size 64
                                request-sent true
                                out-interface ethernet-1/31.1
                                reply {
                                    received true
                                    reply-sender 10.20.1.6
                                    udp-data-length 40
                                    mpls-ttl 255
                                    round-trip-time 83
                                    return-code replying-router-is-egress-for-fec-at-stack-depth-n
                                    return-subcode 1
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

4.2.1.4.2 Performing an LSP trace to an uncolored SR-MPLS TE policy

Procedure

To trace the path of a SR-TE tunnel using an uncolored SR-MPLS TE policy, execute the **tools oam lsp-trace te-policy sr-uncolored policy** command, specifying the uncolored SR-MPLS TE policy name and the protocol origin. To display the results, use the **info from state oam lsp-trace te-policy sr-uncolored policy protocol-origin session-id** command, specifying the session ID output from the **lsp-trace**.

Example: Perform an LSP trace to an uncolored SR-MPLS TE policy

```
--{ + running }--[  ]--
# tools oam lsp-trace te-policy sr-uncolored policy polABCEF protocol-origin local
/:
Initiated LSP Trace for TE-policy polABCEF with session id 49153.
Please check "info from state oam" for result
```

Example: Display results of the LSP trace

```
--{ + running }--[  ]--
# info from state oam lsp-trace te-policy sr-uncolored policy polABCEF protocol-origin local session-
id 49153
    oam {
        lsp-trace {
            te-policy {
                sr-uncolored {
                    policy polABCEF protocol-origin local {
                        session-id 49153 {
                            test-active false
                            path-destination {
                                ip-address 127.0.0.1
                            }
                            hop 1 {
                                probe 1 {
                                    probe-size 188
                                    probes-sent 1
                                    reply {
```

```
                                    received true
                                    reply-sender 10.20.1.2
                                    udp-data-length 32
                                    mpls-ttl 1
                                    round-trip-time 165172
                                    return-code replying-router-is-egress-for-fec-at-stack-depth-n
                                    return-subcode 4
                                }
                            }
                            probe 2 {
                                probe-size 156
                                probes-sent 1
                                reply {
                                    received true
                                    reply-sender 10.20.1.2
                                    udp-data-length 68
                                    mpls-ttl 1
                                    round-trip-time 54673
                                    return-code label-switched-at-stack-depth-n
                                    return-subcode 3
                                }
                                downstream-detailed-mapping 1 {
                                    mtu 1500
                                    address-type ipv4-numbered
                                    downstream-router-address 10.10.3.3
                                    downstream-interface-address 10.10.3.3
                                    mpls-label 1 {
                                        label IMPLICIT_NULL
                                        protocol isis
                                    }
                                    mpls-label 2 {
                                        label 70019
                                        protocol isis
                                    }
                                    mpls-label 3 {
                                        label 70009
                                        protocol isis
                                    }
                                }
                            }
                        }
                        hop 2 {
                            probe 1 {
                                probe-size 156
                                probes-sent 1
                                reply {
                                    received true
                                    reply-sender 10.20.1.3
                                    udp-data-length 32
                                    mpls-ttl 2
                                    round-trip-time 103751
                                    return-code replying-router-is-egress-for-fec-at-stack-depth-n
                                    return-subcode 3
                                }
                            }
                            probe 2 {
                                probe-size 124
                                probes-sent 1
                                reply {
                                    received true
                                    reply-sender 10.20.1.3
                                    udp-data-length 64
                                    mpls-ttl 2
                                    round-trip-time 8262
```

```
                                    return-code label-switched-at-stack-depth-n
                                    return-subcode 2
                                }
                                downstream-detailed-mapping 1 {
                                    mtu 1500
                                    address-type ipv4-numbered
                                    downstream-router-address 10.10.5.5
                                    downstream-interface-address 10.10.5.5
                                    mpls-label 1 {
                                        label IMPLICIT_NULL
                                        protocol isis
                                    }
                                    mpls-label 2 {
                                        label 70009
                                        protocol isis
                                    }
                                }
                            }
                        }
                        hop 3 {
                            probe 1 {
                                probe-size 124
                                probes-sent 1
                                reply {
                                    received true
                                    reply-sender 10.20.1.5
                                    udp-data-length 32
                                    mpls-ttl 3
                                    round-trip-time 57971
                                    return-code replying-router-is-egress-for-fec-at-stack-depth-n
                                    return-subcode 2
                                }
                            }
                            probe 2 {
                                probe-size 92
                                probes-sent 1
                                reply {
                                    received true
                                    reply-sender 10.20.1.5
                                    udp-data-length 60
                                    mpls-ttl 3
                                    round-trip-time 101694
                                    return-code label-switched-at-stack-depth-n
                                    return-subcode 1
                                }
                                downstream-detailed-mapping 1 {
                                    mtu 1500
                                    address-type ipv4-numbered
                                    downstream-router-address 10.10.10.6
                                    downstream-interface-address 10.10.10.6
                                    mpls-label 1 {
                                        label IMPLICIT_NULL
                                        protocol isis
                                    }
                                }
                            }
                        }
                        hop 4 {
                            probe 1 {
                                probe-size 92
                                probes-sent 1
                                reply {
                                    received true
                                    reply-sender 10.20.1.6
```

```
                                    udp-data-length 32
                                    mpls-ttl 4
                                    round-trip-time 14393
                                    return-code replying-router-is-egress-for-fec-at-stack-depth-n
                                    return-subcode 1
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

4.3 Bidirectional Forwarding Detection

BFD is a lightweight mechanism used to monitor the liveliness of a remote neighbor. It is lightweight enough so that the ongoing sending and receiving mechanism can be implemented in the forwarding hardware. Because of this lightweight nature, BFD can send and receive messages at a much higher rate than other control plane hello mechanisms providing faster detection of connection failures.

SR Linux supports BFD asynchronous mode, where BFD control packets are sent between two systems to activate and maintain BFD neighbor sessions between them.

BFD can be configured to monitor connectivity for the following:

- BGP peers – see Configuring BFD under the BGP protocol
- next hops for static routes – see Configuring BFD for static routes
- OSPF adjacencies – see Configuring BFD under OSPF
- IS-IS adjacencies – see Configuring BFD under IS-IS
- link layer LDP adjacencies - see Configuring BFD on an LDP interface

SR Linux supports one BFD session per port/connector, or up to 1152 sessions for an eight slot chassis, depending on the hardware configuration.

On SR Linux systems that support link aggregation groups (LAGs), SR Linux supports micro-BFD, where BFD sessions are established for individual members of a LAG. If the BFD session for one of the links indicates a connection failure, the link is taken out of service from the perspective of the LAG. See Micro-BFD.

4.3.1 Configuring BFD for a subinterface

Procedure

You can enable BFD with an associated subinterface and set values for intervals and criteria for declaring a session down.

Timer values are in microseconds. The detection interval for the BFD session is calculated by multiplying the value of the negotiated transmission interval by the value specified in this field.

Example

The following example configures BFD for a subinterface.

```
--{ candidate shared default }--[  ]--
# info bfd
    bfd {
        subinterface ethernet-1/2.1 {
            admin-state enable
            desired-minimum-transmit-interval 250000
            required-minimum-receive 250000
            detection-multiplier 3
        }
    }
```

4.3.2 Configuring BFD under the BGP protocol

Procedure

You can configure BFD under the BGP protocol at the global, group, or neighbor level.

Before enabling BFD, you must first configure it for a subinterface and set timer values. See Configuring BFD for a subinterface.

Example: Configure BFD under the BGP protocol at the global level

```
--{ candidate shared default }--[  ]--
# info network-instance default
    network-instance default {
        protocols {
            bgp {
                failure-detection {
                    enable-bfd true
                    }
                }
            }
        }
    }
```

Example: Configure BFD for a BGP peer group

The following example configures BFD for the links between peers within an associated BGP peer group.

```
--{ * candidate shared default }--[  ]--
# info network-instance default protocols bgp
    network-instance default {
        protocols {
            bgp {
                group test {
                    failure-detection {
                        enable-bfd true
                    }
                }
            }
        }
    }
```

Example: Configure BFD for BGP neighbors

The following example configures BFD for the link between BGP neighbors.

```
--{ candidate shared default }--[  ]--
# info network-instance default protocols bgp
    network-instance default {
        protocols {
            bgp {
                neighbor 192.168.0.1 {
                    failure-detection {
                        enable-bfd true
                        fast-failover true
                    }
                }
            }
        }
    }
```

4.3.3 Configuring BFD for static routes

Procedure

You can use BFD as a failure detection mechanism for monitoring the reachability of next hops for static routes. When BFD is enabled for a static route, it makes an active BFD session between the local router and the defined next hops required as a condition for a static route to be operationally active.

You enable BFD for specific next-hop groups; as a result, BFD is enabled for any static route that refers to the next-hop group. If multiple next hops are defined within the next-hop group, a BFD session is established between the local address and each next hop in the next-hop group.

A static route is considered operationally up if at least one of the configured next-hop addresses can establish a BFD session. If the BFD session fails, the associated next hop is removed from the FIB as an active next hop.

Example

The following example enables BFD for a static route next hop:

```
--{ * candidate shared default }--[ network-instance black ]--
# info next-hop-groups
    next-hop-groups {
        group static-ipv4-grp {
            admin-state enable
            nexthop 1 {
                failure-detection {
                    enable-bfd {
                        local-address 192.0.2.1
                    }
                }
            }
        }
    }
```

A BFD session is established between the address configured with the **local-address** parameter and each next-hop address before that next-hop address is installed in the forwarding table.

All next-hop BFD sessions share the same timer settings, which are taken from the BFD configuration for the subinterface where the address in **local-address** parameter is configured. See Bidirectional Forwarding Detection.

4.3.4 Configuring BFD under OSPF

Procedure

For OSPF and OSPFv2, you can enable BFD at the interface level to monitor the connectivity between the router and its attached network.

Example

```
--{ candidate shared default }--[  ]--
# info network-instance default protocols ospf
    network-instance default {
        interface ethernet-1/1.1 {
            interface-ref {
                interface ethernet-1/1
                subinterface 1
            }
        }
        protocols {
            ospf {
                instance o1 {
                    version ospf-v2
                    area 1.1.1.1 {
                        interface ethernet-1/1.1 {
                            failure-detection {
                                enable-bfd true
                            }
                        }
                    }
                }
            }
        }
    }
```

4.3.5 Configuring BFD under IS-IS

Procedure

You can configure BFD at the interface level for IS-IS. You can optionally configure a BFD-enabled TLV to be included for IPv4 or IPv6 on the IS-IS interface.

Example

```
--{ candidate shared default }--[  ]--
# info network-instance default protocols isis
    network-instance default {
        interface ethernet-1/1.1 {
            interface-ref {
                interface ethernet-1/1
                subinterface 1
            }
        }
        protocols {
```

```
            isis {
                instance i1 {
                    ipv4-unicast {
                        admin-state enable
                    }
                    interface ethernet-1/1.1 {
                        ipv4-unicast {
                            enable-bfd true
                            include-bfd-tlv true
                        }
                    }
                }
            }
        }
    }
```

4.3.6 Configuring BFD on an LDP interface

Procedure

You can configure BFD on an IPv4 or IPv6 LDP interface.

Example

This example enables BFD on an LDP interface.

```
--{ +* candidate shared default }--[  ]--
# info network-instance default protocols ldp
    network-instance default {
        protocols {
            ldp {
                dynamic-label-block d1
                discovery {
                    interfaces {
                        hello-holdtime 30
                        hello-interval 10
                        interface ethernet-1/1.1 {
                            ipv4 {
                                admin-state enable
                                enable-bfd true
                            }
                            ipv6 {
                                admin-state enable
                                enable-bfd true
                            }
                        }
                    }
                }
            }
        }
    }
```

4.3.7 Viewing the BFD state

Procedure

Use the **info from state** command to verify the BFD state for a network-instance.

Example

```
# info from state bfd network-instance default peer 30
    bfd {
        network-instance default {
            peer 30 {
                oper-state up
                local-address 192.168.1.5
                remote-address 192.168.1.3
                remote-discriminator 25
                subscribed-protocols bgp_mgr
                session-state UP
                remote-session-state UP
                last-state-transition 2020-01-24T16:22:55.224Z
                failure-transitions 0
                local-diagnostic-code NO_DIAGNOSTIC
                remote-diagnostic-code NO_DIAGNOSTIC
                remote-minimum-receive-interval 1000000
                remote-control-plane-independent false
                active-transmit-interval 250000
                active-receive-interval 250000
                remote-multiplier 3
                async {
                    last-packet-transmitted 2020-01-24T16:23:19.385Z
                    last-packet-received 2020-01-24T16:23:18.906Z
                    transmitted-packets 32
                    received-packets 32
                    up-transitions 1
                }
            }
        }
    }
```

4.4 Micro-BFD

Micro-BFD refers to running BFD over the individual links in a LAG to monitor the bidirectional liveliness of the Ethernet links that make up the LAG.

A LAG member cannot be made operational within the LAG until the micro-BFD session is fully established. If a micro-BFD session fails, the corresponding Ethernet link is taken out of service from the perspective of the LAG.

Micro-BFD is supported on Ethernet LAG interfaces with an IP interface. Micro-BFD sessions are associated with each individual link. When enabled, the state of the individual links depends on the micro-BFD session state:

- Micro-BFD sessions must be established between both endpoints of a link before the link can be operationally up.
- If the micro-BFD session fails, the associated Ethernet link becomes operationally down from the perspective of the LAG.
- If LACP is not enabled for the LAG and the Ethernet port is up, the system attempts to re-establish the micro-BFD session with the far end of the link.
- If LACP enabled for the LAG and the Ethernet port is up, the system attempts to re-establish the micro-BFD session with the far end of the link when LACP reaches distributing state.

If a link is not active for forwarding from the perspective of a LAG, ARP can still be performed across the link. For example, when a link is being brought up, and its micro-BFD session is not yet established, ARP can still be performed for the MAC address at the far end of the link, even though the link is not yet part of the LAG.

Micro-BFD packets bypass ingress and egress subinterface/interface ACLs, but received micro-BFD packets can be matched by CPM filters for filtering and logging.

Micro-BFD is supported on all SR Linux systems that also support LAGs: 7250 IXR; 7250 IXR-X, 7220 IXR-D1, D2, and D3; 7220 IXR-H2 and H3.

4.4.1 Configuring micro-BFD for a LAG interface

Procedure

To configure micro-BFD for a LAG interface, you configure IP addresses to be used as the source address for IP packets and a remote address for the far end of the BFD session.

You can specify the minimum interval in microseconds between transmission of BFD control packets, as well as the minimum acceptable interval between received BFD control packets. The detection-multiplier setting specifies the number of packets that must be missed to declare the BFD session as down.

Example

```
--{ * candidate shared default }--[  ]--
# info bfd micro-bfd-sessions
    micro-bfd-sessions {
        lag-interface lag1 {
            admin-state enable
            local-address 192.35.2.5
            remote-address 192.35.2.3
            desired-minimum-transmit-interval 250000
            required-minimum-receive 250000
            detection-multiplier 3
        }
    }
```

4.4.2 Viewing the micro-BFD state

Procedure

Use the **info from state** command to verify the micro-BFD state for members of a LAG interface.

Example

```
# info from state micro-bfd-sessions lag-interface lag1 member-interface ethernet 2/1
    micro-bfd-sessions
        lag-interface lag1 {
            admin-state UP
            local-address 192.0.2.5
            remote-address 192.0.2.3
            desired-minimum-transmit-interval 250000
            required-minimum-receive 250000
            detection-multiplier 3
            member-interface ethernet 2/1 {
                session-state UP
                remote-session-state UP
```

```
                last-state-transition 2020-01-24T16:22:55.224Z
                last-failure-time 2020-01-24T16:22:55.224Z
                failure-transitions 0
                local-discriminator 25
                remote-discriminator 25
                local-diagnostic-code NO_DIAGNOSTIC
                remote-diagnostic-code NO_DIAGNOSTIC
                remote-minimum-receive-interval 1000000
                remote-control-plane-independent false
                active-transmit-interval 250000
                active-receive-interval 250000
                remote-multiplier 3
                async {
                   last-clear 2020-01-23T16:21:19.385Z
                   last-packet-transmitted 2020-01-24T16:23:19.385Z
                   last-packet-received 2020-01-24T16:23:18.906Z
                   transmitted-packets 32
                   received-errored-packets 3
                   received-packets 32
                   up-transitions 1
                }
             }
          }
       }
```

4.5 Seamless Bidirectional Forwarding Detection (S-BFD)

Overview

BFD detects connection failures faster than other hello mechanisms. However, if many BFD sessions are configured to detect links, very long negotiation times result in reduced system performance. You can configure seamless bidirectional forwarding detection (S-BFD), which is a simplified mechanism that speeds up a BFD session by eliminating the negotiation and state establishment process. This is accomplished primarily by predetermining the session discriminator and using specific mechanisms to distribute the discriminators to a remote network entity. This allows client applications or protocols to quickly initiate and perform connectivity tests. A per-session state is maintained only at the head-end of a session. The tail-end reflects the BFD control packets back to the head-end.

Initiator and reflector

An S-BFD session is established between an initiator and a reflector. SR Linux supports only one instance of a reflector in each node. A discriminator is assigned to initiator and reflector.

The initiator initiates an S-BFD session on a network node and performs a continuity test by sending S-BFD packets to the reflector. The reflector receives the S-BFD packet and reflects the S-BFD packet back along with the state value based on its current state.

The following information is swapped in the S-BFD response:

- The source and destination IP addresses
- The source and destination UDP ports
- The initiator and reflector discriminators

See Configuring an S-BFD reflector for information about how to configure a reflector. An SR Linux router can be both an initiator and a reflector, thereby allowing you to configure different S-BFD sessions.

S-BFD discriminator

SR Linux supports the following methods of mapping an S-BFD remote IP address with its discriminator:

- Static configuration
- Automatic learning using opaque IS-IS routing extensions

You can statically configure an S-BFD remote IP address and discriminator for each network instance. The S-BFD initiator immediately starts sending S-BFD packets if the discriminator value of the far-end reflector is known. A session set up is not required. The **INIT** state is not present in an S-BFD session. The initiator state changes from **AdminDown** to **Up** when it begins to send S-BFD packets. The following table lists the S-BFD packet information that the initiator sends to the reflector.

Table 5: Fields in S-BFD packet

Source IP address	This is the local session IP address. For IPv6, this is a global unicast address belonging to the node.
Destination IP address	This is the IP address of the reflector, and it needs to be configured.
My discriminator	This is the locally assigned discriminator.
Your discriminator	This is the discriminator value of the reflector, and it needs to be configured.

See Statically configuring an S-BFD discriminator for more information about how to configure an S-BFD discriminator.

If the initiator receives a valid response from the reflector with an **Up** state, the initiator declares the S-BFD session state as **Up**. If the initiator fails to receive a specific number of responses, as determined by the BFD multiplier in the BFD template for the session, the initiator declares the S-BFD session state as **Failed**. If any of the discriminators change, the session fails and the router attempts to restart with the new values. If the reflector discriminator changes at the far-end peer, the session fails. The mapping may not have been updated locally before the system checks for a new reflector discriminator from the local mapping table. Therefore the session is bounced and brought up with the new values. If any discriminator is deleted, the corresponding S-BFD sessions are deleted.

SR Linux supports automatic mapping of an S-BFD remote IP address with its discriminator using the IS-IS protocol extensions. The IS-IS protocol uses a sub-TLV of the capabilities TLV to advertise and distribute discriminators. See Automatically mapping an S-BFD discriminator for more information.

Routed and controlled return path

S-BFD supports the following forms of returning transmitted S-BFD packets back to the initiator:

- Routed return
- Controlled return path

In routed return, S-BFD uses an initiator-reflector model where an initiator sends S-BFD messages to a reflector using the discriminator of the reflectors. The reflector reflects the S-BFD message back to the initiator via IPv4 or IPv6 routing.

In controlled return path for SR-Policy, the initiating node embeds a SID, typically a binding SID that is used by the reflecting node, to determine the correct path back to the initiator. The S-BFD message is then forwarded to a path that is identical or similar to the original path that the message was sent by the initiator.

S-BFD state

S-BFD session state is reported at the network instance, policy, and system levels. See Viewing the S-BFD state for more information.

4.5.1 Statically configuring an S-BFD discriminator

Procedure

To statically map an S-BFD remote IP address with its discriminator for each network instance, you configure the **network-instance bfd seamless-bfd** command and specify the peer IP address and discriminator.

Example: Statically configuring an S-BFD discriminator

This is an example for statically configuring an S-BFD discriminator.

```
--{ + candidate shared default }--[  ]--
# info network-instance default bfd seamless-bfd
    network-instance default {
        bfd {
            seamless-bfd {
                peer 192.0.2.0 {
                    discriminator 30
                }
            }
        }
    }
```

4.5.2 Automatically mapping an S-BFD discriminator

Procedure

SR Linux supports automatic mapping of an S-BFD remote IP address with its discriminator using IGP routing protocol extensions. The IS-IS protocol uses a sub-TLV of the capabilities TLV to distribute S-BFD discriminators. There is no explicit configuration to enable or disable router capability advertisement.

Example: Output from BFD state

This example shows an output of an automatically mapped S-BFD discriminator.

```
--{ + running }--[  ]--
# info from state bfd
    bfd {
        total-bfd-sessions 2
        total-unmatched-bfd-packets 1
        network-instance base {
            peer 16385 {
                oper-state up
                local-address 1.1.1.3
                remote-address 127.0.64.1
                remote-discriminator 524289
                subscribed-protocols SRPOLICY
                session-state UP
                remote-session-state UP
                last-state-transition "2024-05-15T19:15:58.117Z (49 seconds ago)"
```

```
                failure-transitions 0
                local-diagnostic-code NO_DIAGNOSTIC
                remote-diagnostic-code NO_DIAGNOSTIC
                remote-minimum-receive-interval 1000000
                remote-control-plane-independent false
                active-transmit-interval 1000000
                active-receive-interval 1000000
                remote-multiplier 3
                te-policy-name C_to_Fipv4
                te-policy-segment-list-index 1
                te-policy-protocol-origin LOCAL
                te-policy-segment-list-lsp-index 216
                sr-policy-endpoint 1.1.1.6
                async {
                    last-packet-transmitted "2024-05-15T19:16:43.140Z (4 seconds ago)"
                    last-packet-received "2024-05-15T19:16:43.146Z (4 seconds ago)"
                    transmitted-packets 61
                    received-packets 61
                    up-transitions 1
                }
            }
            peer 16386 {
                oper-state up
                local-address 1.1.1.3
                remote-address 127.0.64.2
                remote-discriminator 524289
                subscribed-protocols SRPOLICY
                session-state UP
                remote-session-state UP
                last-state-transition "2024-05-15T19:15:58.119Z (49 seconds ago)"
                failure-transitions 0
                local-diagnostic-code NO_DIAGNOSTIC
                remote-diagnostic-code NO_DIAGNOSTIC
                remote-minimum-receive-interval 1000000
                remote-control-plane-independent false
                active-transmit-interval 1000000
                active-receive-interval 1000000
                remote-multiplier 3
                te-policy-name C_to_Fipv4
                te-policy-segment-list-index 2
                te-policy-protocol-origin LOCAL
                te-policy-segment-list-lsp-index 217
                sr-policy-endpoint 1.1.1.6
                async {
                    last-packet-transmitted "2024-05-15T19:16:43.651Z (4 seconds ago)"
                    last-packet-received "2024-05-15T19:16:43.695Z (4 seconds ago)"
                    transmitted-packets 62
                    received-packets 62
                    up-transitions 1
                }
            }
        }
    }
```

4.5.3 Configuring an S-BFD reflector

Procedure

To enable and configure an S-BFD reflector, use the **network-instance bfd seamless-bfd reflector** command. You must allocate the discriminator value from the S-BFD reflector pool that ranges from 524288 to 526335.

Note:
Only a single reflector discriminator is supported for each network instance.

Example: Configuring an S-BFD reflector

The following example configures an S-BFD reflector.

```
--{ + candidate shared default }--[  ]--
# info network-instance default bfd seamless-bfd reflector abc
    network-instance default {
        bfd {
            seamless-bfd {
                reflector abc {
                    local-discriminator 524289
                    admin-state enable
                    description test
                }
            }
        }
    }
```

4.5.4 Viewing the S-BFD state

Procedure

Use the **info from state** command to verify the S-BFD state.

Example: Viewing the S-BFD state at network instance level

The following example displays the S-BFD status at the network instance level.

```
--{ + running }--[  ]--
# info from state network-instance base bfd seamless-bfd
    network-instance base {
        bfd {
            seamless-bfd {
                peer 1.1.1.6 {
                    discriminator 524289
                }
                reflector 1.1.1.3 {
                    local-discriminator 524289
                    admin-state enable
                }
            }
        }
    }
```

Example: Viewing the S-BFD state at the policy level

The following example displays the S-BFD status at the policy level.

```
--{ + running }--[  ]--
# info from state network-instance base maintenance-policies policy mp
    network-instance base {
        maintenance-policies {
            policy mp {
                revert-timer disable
                seamless-bfd {
                    detection-multiplier 3
                    desired-minimum-transmit-interval 1000000
                    hold-down-timer 4
                    wait-for-up-timer 3
                    mode linear
                    threshold 1
                }
            }
        }
    }
```

Example: Viewing the S-BFD state at the system level

The following example displays the S-BFD status at the system level.

```
--{ + running }--[  ]--
# info from state bfd
    bfd {
        total-bfd-sessions 2
        total-unmatched-bfd-packets 1
        network-instance base {
            peer 16385 {
                oper-state up
                local-address 1.1.1.3
                remote-address 127.0.64.1
                remote-discriminator 524289
                subscribed-protocols SRPOLICY
                session-state UP
                remote-session-state UP
                last-state-transition "2024-05-15T19:15:58.117Z (49 seconds ago)"
                failure-transitions 0
                local-diagnostic-code NO_DIAGNOSTIC
                remote-diagnostic-code NO_DIAGNOSTIC
                remote-minimum-receive-interval 1000000
                remote-control-plane-independent false
                active-transmit-interval 1000000
                active-receive-interval 1000000
                remote-multiplier 3
                te-policy-name C_to_Fipv4
                te-policy-segment-list-index 1
                te-policy-protocol-origin LOCAL
                te-policy-segment-list-lsp-index 216
                sr-policy-endpoint 1.1.1.6
                async {
                    last-packet-transmitted "2024-05-15T19:16:43.140Z (4 seconds ago)"
                    last-packet-received "2024-05-15T19:16:43.146Z (4 seconds ago)"
                    transmitted-packets 61
                    received-packets 61
                    up-transitions 1
                }
            }
            peer 16386 {
```

```
                oper-state up
                local-address 1.1.1.3
                remote-address 127.0.64.2
                remote-discriminator 524289
                subscribed-protocols SRPOLICY
                session-state UP
                remote-session-state UP
                last-state-transition "2024-05-15T19:15:58.119Z (49 seconds ago)"
                failure-transitions 0
                local-diagnostic-code NO_DIAGNOSTIC
                remote-diagnostic-code NO_DIAGNOSTIC
                remote-minimum-receive-interval 1000000
                remote-control-plane-independent false
                active-transmit-interval 1000000
                active-receive-interval 1000000
                remote-multiplier 3
                te-policy-name C_to_Fipv4
                te-policy-segment-list-index 2
                te-policy-protocol-origin LOCAL
                te-policy-segment-list-lsp-index 217
                sr-policy-endpoint 1.1.1.6
                async {
                    last-packet-transmitted "2024-05-15T19:16:43.651Z (4 seconds ago)"
                    last-packet-received "2024-05-15T19:16:43.695Z (4 seconds ago)"
                    transmitted-packets 62
                    received-packets 62
                    up-transitions 1
                }
            }
        }
    }
```

5 OAM monitoring and reporting

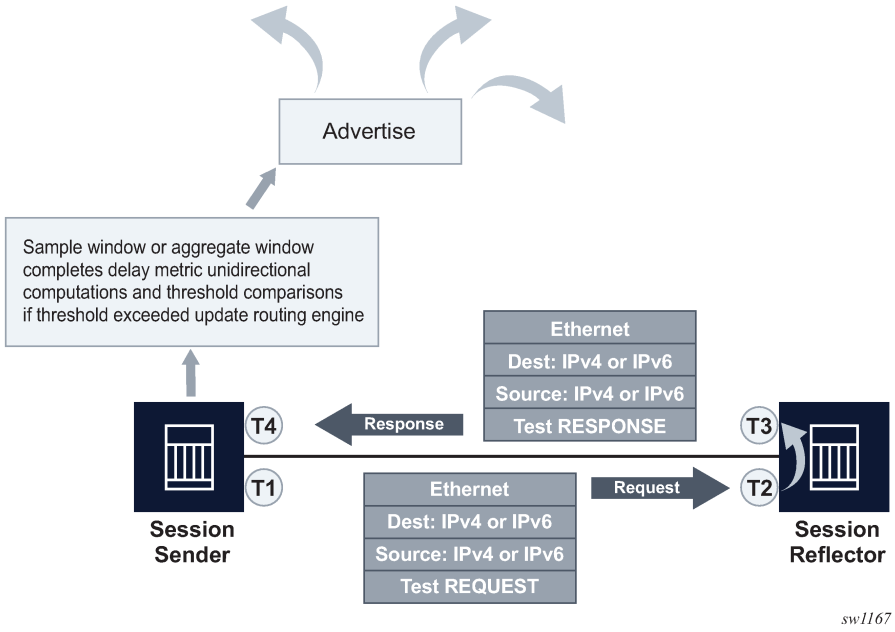
OAM fault and performance tools monitor and report information about the network infrastructure and the services that rely on that infrastructure.

5.1 Link measurement

Network elements use routing protocols to exchange information about local links, which can influence routing decisions. These interface attributes are typically static in nature. By using tools specifically designed to measure IP performance, dynamic unidirectional delay can be included in the advertised link attributes. A link measurement test is one such method for measuring and reporting delay information for directly connected IP peers. Link measurement uses the STAMP protocol defined in RFC 8762 to measure delay for an IP interface

The following figure shows directly connected IP interfaces and the link measurement interaction with routing.

Figure 4: Link measurement interactions



5.1.1 Link measurement template

Creation of a link measurement template is the first step of a link measurement test. The link measurement template is created by configuring the common test parameters using the **oam link-measurement**

command. After the measurement template is created, an SR Linux interface references the link measurement template using the **oam link-measurement interface dynamic-measurement link-measurement-template** command. When the association between the interface and the template is established, the interface executes a process to determine the operational state of the test and detect any defect conditions that may prevent test execution. If no underlying conditions are present, the IP interface delay measurements are collected in measurement windows, compared with the configured thresholds, and reported to the routing engine for further processing when required.

Link measurement template parameters

The parameters that are configured using the **oam link-measurement interface dynamic-measurement link-measurement-template** command define the test criteria used by the test. Conceptually, the test criteria are divided into the following groups:

- General configuration
- Collection and reporting
- Protocol

Modifying a link measurement template

SR Linux supports the modification of active link measurement templates. That includes administratively disabling a link measurement template that IP interfaces are actively referencing. Modifying existing parameters causes interface delay tests that reference the modified template to terminate the current sample, and aggregate measurement windows, and start new measurement windows using the updated template parameters. The previous historical results are maintained, but the state field of the measurement window coinciding with the change indicates **Terminated**. Changing the description or the **last-reported-delay-hold** configuration does not cause a termination of the current sample and aggregate measurement windows.

Deleting a link measurement template

A link measurement template cannot be deleted if interfaces are referencing that template.

5.1.1.1 General configuration

The general configurations included in the **oam link-measurement measurement-template** command influence probe frequency, the delay metric type to monitor, and retention of the delay measurement last reported.

- The probe frequency, configured using the **interval** command, defines the transmission rate of the test packet.
- The **delay** command configures the delay metric (minimum, maximum, or average) that is used for comparison against any configured thresholds. This metric is the same for both types of measurement windows, the **sample-window** and **aggregate-sample-window**.
- The **unidirectional-measurement** command specifies the method used to compute the unidirectional delay. If the clock synchronization between nodal clocks used by the OAM timestamp function is not synchronized to near exact accuracy, the **derived** option must be used. Specifying this option calculates the unidirectional measurement using the round trip delay divided by two computation. If synchronization can meet near exact accuracy, the **actual** option can be used. Specifying this option calculates the forward delay using the forward direction timestamps, T2-T1 computation.

- When the operational state of the link measurement test transitions to down, the OAM function instructs the routing engine to clear the last reported delay value at the expiration of the **last-reported-delay-hold** value. A previously reported delay is considered valid for the duration of this period and is cleared if the timer reaches zero. If the operational state returns to up before the timer expires, no action is taken to clear the previous value. The counter is reset to the configured value, waiting for the next operational down event. If the **last-reported-delay-timer** is set to zero, previously reported delay values from that test are cleared when the operational state changes to down without any additional time.

  An operational state is up for a test if:

  – the measurement template is administratively up

  – the source UDP port is available

  – an IPv4 or IPv6 protocol is administratively up

  – there are no internal errors that prevent the test from initializing

- The aging timer does not start a count to zero for failure conditions that do not affect the interface delay test operational state. The delay measurement last reported is maintained when conditions external to the interface delay test, such as fault conditions on the port, IP interface, routing changes, and so on, occur.

5.1.1.2 Collection and reporting

The collection and reporting parameters define the following:

- length of the **sample-window** and **aggregate-sample-window**. Two measurement windows are provided to support use cases that require a reporting hierarchy and both include the same configuration options.
- thresholds that trigger reporting. The threshold values determine when the measurement window updates the reported delay value.

Multiplier

The measurement windows use the **multiplier** command to determine the length of time that the measurement window remains open. The sample window length is multiples of the interval. This window stores the results of individual test probes for a total length of the **interval** multiplied by the **multiplier** value. The aggregate sample window multiplier length is the number of sample windows. This window stores the number of results passed from individual sample windows. In the aggregate sample window, the minimum, maximum, and average calculations are based on the results received from the sample window. For example, if the delay metric of interest is the average, the aggregate sample is a collection of averages passed from the sample window. The reporting in the aggregate sample window is as follows:

- The minimum is the minimum value for all the averages received.
- The maximum is the maximum value from all the averages received.
- The average is the average of all the averages received.

Window integrity

The comparison to thresholds and reporting decisions occurs at the end of the measurement window if it completes without termination and is deemed integral based on the **window-integrity** command configuration. Integrity is a percentage-based calculation that determines the number of samples that must

be present in the measurement window for that window to be considered integral. If the number of samples in the window equals or exceeds the number of required samples, the result is treated as representative and follows normal post-measurement window processing. However, if the number of samples in the window does not achieve integrity, the result is not considered representative and is only recorded for historical purposes, but is otherwise ignored and not processed. By default, integrity checking is disabled and all results from a measurement window are treated as integral and compared to the configured thresholds.

Threshold

There are two types of thresholds:

- a microsecond increase or decrease, configured using the **absolute** command
- a percentage increase or decrease, configured using the **relative** command

Thresholding compares the measurement window result to the delay measurement last reported at the end of the successful (completed) measurement window. Reporting is on a per-threshold, per-measurement window basis. If multiple thresholds are reached for a completed measurement window, only one threshold triggers an update to the routing engine.

Configuration of the measurement windows depends on the specific solution requirements. The two measurement windows collect information regardless of configured thresholds. Both types of measurement windows support their own threshold and integrity configuration. By default, thresholds for both measurement windows are disabled; that is, neither window can report any values to the routing engine.

Reporting

Reporting is enabled by default and the routing engine is informed of threshold events. The notification process uses the **reporting** command and threshold configurations. At least one threshold must be configured to report to the routing engine. Disabling reporting allows the function to execute but prevents the reporting of threshold events to the routing engine. If this value is toggled and a value was previous reported, the reported value is cleared, and the process returns to the initial reporting phase.

5.1.1.3 Protocol

The **oam link-measurement measurement-template stamp** command parameters influence the format of the test packet, processing, QoS handling, IPv6 discovery, and return path.

Source UDP port

By default, link measurement uses dynamic source UDP ports. However, a specific source UDP port can be configured if required, from a range of source UDP ports allocated to STAMP which is, 64374 to 64383. The UDP port must previously be allocated to the link measurement application. See Allocating source UDP port to link measurement for more information about steps to allocate a source UDP port to the link measurement application.

IPv6 destination discovery

IPv6 destination address discovery allows the discovery of a single directly connected IPv6 peer. When this option is enabled, a bootstrap function using an ICMPv6 echo request with a destination ff02::2 is generated. When the directly-connected peer responds, the link measurement function uses the source address of the ICMPv6 echo response as the destination address for the link measurement test packets.

The process has four main components:

- Enabling the functions using **admin-state** command.
- Configuring the **discovery-interval** command. This is the initial timer used by the discovery process to discover the peer. This interval is used for the duration of the **discovery-timer**.
- Implementing the discovery phase. If the timer expires or the peer is discovered before the expiration of the **discovery-timer**, the process reverts back to the **update-interval**.
- Implementing the **update-interval**. This is an optional maintenance component of the peer address that runs at a slower rate. This option is not required and can be disabled in environments where the peer address is unlikely to change. If the peer is not discovered during the **discovery-timer** and with the **update-interval** disabled, the peer fails to be discovered. Disable and then enable the IPv6 protocol to restart the discovery process.

Return path

By default, the session reflectors use routing to return the response packet to the session sender. There are instances when it may be beneficial to be selective about the IP interface used for the return path. For example, when multiple tests are executed on different interfaces between the same pair of nodes, and using non-directly connected interface addresses, and ECMP exists between the two nodes. In this case, the **return-path link** command can be configured as **true**. This includes the return path TLV and link sub TLV in the test packet. This configuration instructs the session reflector to send the response out to the same IP interface on which it was received. The destination IP address for the response packet must be installed in the forwarding table and reachable from that interface. If the routing engine determines that the prefix is not reachable from that interface, the response packet is dropped at the reflector.

5.1.2 Interface assignment

The test criteria-specific link measurement is configured in the link measurement template. The delay test is executed from the network instance with the type default and requires and interface that is part of the network instance **oam link-measurement interface** command. The link measurement template does not include interface-specific requirements, such as the IP protocol encapsulating the test packet or IP source and destination addressing.

5.1.2.1 IP addressing

To enable dynamic measurements for the interface, configure a link measurement template and enable the test protocol using the **oam link-measurement interface dynamic-measurement stamp ipv4** or **oam link-measurement interface dynamic-measurement stamp ipv6** command. Only one protocol, IPv4 or IPv6, can be enabled for an interface delay test at any time. Interfaces defined as **loopback** do not support interface delay tests and are an invalid interface type.

IPv4 address auto-discovery

When the IPv4 protocol is enabled with no addressing configured, the source address is automatically assigned to the primary IPv4 address of the IP interface. The destination address is automatically assigned if the primary IPv4 address has a prefix length of 30 or 31. In other cases, such as shorter prefix lengths or unnumbered interfaces, the destination address cannot be resolved and must be configured manually. The **source-ip** and **destination-ip** commands take precedence over the auto-assigned addressing; the IPv4 addresses must be unicast.

IPv4 auto-assigned addressing is not updated for operationally up interface delay tests when the IP addressing associated with that interface is changed. Nokia suggests the following options to update the auto-assigned addressing:

- administratively disable and enable the protocol used for the interface delay test
- disable and enable the IP interface under which the IP address has changed

IPv6 address auto-discovery

When the IPv6 protocol is enabled without any source address, the system uses the link-local address associated with the interface as the source. If there is no destination address configured, the destination discovery process is initiated if the associated link measurement template assigned to this interface has the following command enabled.

```
oam link-measurement measurement-template stamp ipv6-destination-discovery
```

5.1.2.2 Test initialization

When the link measurement template is assigned to an IP interface, the audit process determines the operational state of the test. The interface delay test transitions to operationally up if the following conditions are met:

- the associated measurement template is administratively enabled
- there is an administratively enabled test protocol configured using the **ipv4** or **ipv6** command
- the system resources are available to start the test

Further validation determines if there are any underlying conditions that are considered detectable transmission errors, which are listed in the following table.

Table 6: Detectable transmission errors

Detectable transmit error	Description	Prevents transmission
none	No detectable errors	No
subinterface-down	The link measurement test is configured on an IP interface with an operationally down state	Yes
unexpected-error	Router resources not available	No
no-route	The routing lookup has failed to resolve the destination as reachable from the interface where the test is configured. The packet is transmitted out of the interface resolved by the routing engine.	No
source-ip-not-local	The source IP address configured for the test is not local to the system	No

Detectable transmit error	Description	Prevents transmission
invalid-dest-ip	The destination IP address is not valid. This may occur because of a configuration error or an attempt to use auto-assignment in conditions that are not supported.	Yes
subinterface-type-not-supported	The link measurement test is configured under an interface that does not support these test types (such as loopback interfaces)	Yes
same-source-ip-destination-ip	A configuration error that indicates the source and destination IP addresses are the same	Yes

When all audit conditions successfully pass, the test begins. When no thresholds are configured, the test collects delay information as history, but without at least one configured threshold value, reporting updates to the routing engine are disabled. If at least one threshold is configured, the interface enters the first report phase. Because no previous delay value has been reported, the first measurement window with a configured threshold that completes with integrity triggers the delay measurement report. After this benchmark is set, all subsequent thresholds use the delay measurement last reported as the comparison.

5.1.2.3 History and results

Active interface delay tests retain 50 sample windows and 20 aggregate sample windows in history. The current measurement windows and historical results are not maintained across CPM switchovers. The delay measurement last reported is maintained after a CPM switchover to retain the baseline. The interface does not enter the first reporting phase following a CPM switchover.

The results can be viewed using the **info from state oam link-measurement interface** command.

5.1.3 Allocating source UDP port to link measurement

Procedure

To allocate a source UDP port to link measurement, you specify the UDP port number in the **oam ippm source-udp-port-pools port** command and select **link-measurement** option in the **application-assignment** parameter.

Example: Allocating source USP port to link measurement

This example allocates a source UDP port to perform the link measurement test.

```
--{ + candidate shared default }--[  ]--
# info oam ippm
    oam {
        ippm {
            source-udp-port-pools {
                port 64374 {
```

```
                application-assignment link-measurement
            }
        }
    }
}
```

5.1.4 Performing link measurement test

About this task

Perform the following steps to carry out the link measurement test:

Procedure

Step 1. Create a link measurement template. Perform the following steps:

a. Use the **oam link-measurement measurement-template** command to create a measurement template.

b. Enable the measurement template and configure the parameters as shown in the following example.

Example

Creating a measurement template

```
--{ + candidate shared default }--[  ]--
# info oam link-measurement measurement-template test
    oam {
        link-measurement {
            measurement-template test {
                admin-state enable
                description Test
                unidirectional-measurement derived
                delay average
                interval 1
                last-reported-dynamic-delay-hold 86400
                reporting true
                aggregate-sample-window {
                    multiplier 12
                }
                sample-window {
                    multiplier 10
                }
            }
        }
    }
```

Step 2. Assign an interface to the link measurement template. Perform the following steps:

a. Use the **oam link-measurement interface interface-ref** command to create an interface.

b. Use the **oam link-measurement interface dynamic-measurement link-measurement-template** command to assign the interface to the link measurement template.

c. Use the **oam link-measurement interface dynamic-measurement stamp ipv4** or **oam link-measurement interface dynamic-measurement stamp ipv6** command to configure the protocol and IP address of the source and destination as shown in the following example.

Example

Assigning an interface to the link measurement template

```
--{ + candidate shared default }--[  ]--
# info oam link-measurement interface lm001 dynamic-measurement
    oam {
        link-measurement {
            interface lm001 {
                dynamic-measurement {
                    link-measurement-template test
                    stamp {
                        ipv4 {
                            admin-state disable
                            destination-ip 192.168.0.1
                            source-ip 192.168.0.2
                        }
                        ipv6 {
                            admin-state enable
                            destination-ip 2001:db8::1
                            source-ip 2001:db8::2
                        }
                    }
                }
            }
        }
    }
```

Step 3. When the link measurement template is assigned to an IP interface, the audit process determines the operational state of the test. Further validation determines if there are any underlying conditions that are considered detectable transmission errors. When all audit conditions successfully pass, the delay collection begins.

Step 4. Use the **info from state oam link-measurement interface** command to view the results of the link measurement test as shown in the following example.

Example

Displaying link measurement test results

```
--{ +!* candidate shared default }--[  ]--
# info from state oam link-measurement interface lm001
    oam {
        link-measurement {
            interface lm001 {
                oper-state down
                detectable-transmit-error invalid-dest-ip
                operational-source-address 2001:db8::2
                source-ip-auto-assigned false
                operational-destination-address 2001:db8::1
                destination-ip-auto-assigned false
                in-use-source-udp-port 64374
                in-use-destination-udp-port 862
                stamp-session-sender-id 0
                reporting false
                last-reported-dynamic-delay none
                report-timestamp 1970-01-01T00:00:00.000Z
                report-triggered-by none
                aggregate-newest-index 1
                sample-newest-index 2
                operational-failure [
                    no-protocol
                ]
```

```
                interface-ref {
                    interface ethernet-1/22
                    subinterface 1
                }
                dynamic-measurement {
                    link-measurement-template test
                    stamp {
                        ipv4 {
                            admin-state disable
                            destination-ip 192.168.0.1
                            source-ip 192.168.0.2
                        }
                        ipv6 {
                            admin-state enable
                            destination-ip 2001:db8::1
                            source-ip 2001:db8::2
                        }
                    }
                }
                statistics {
                    aggregate-sample-window {
                        index 1 {
                            end-timestamp-utc 2024-06-13T02:15:56.000Z
                            window-state terminated
                            sample-window-count 1
                            minimum 0
                            maximum 0
                            average 0
                            result 0
                            integrity false
                        }
                    }
                    sample-window {
                        index 1 {
                            end-timestamp-utc 2024-06-13T02:15:55.000Z
                            window-state completed
                            transmitted-packets 10
                            received-packets 8
                            minimum 3128
                            maximum 3828
                            average 3444
                            integrity true
                            error-count 0
                            stamp-unrecognized-flag-count 0
                            stamp-malformed-flag-count 0
                            zero-or-negative-delay-count 0
                            duplicate-packet-count 0
                        }
                        index 2 {
                            end-timestamp-utc 2024-06-13T02:15:56.000Z
                            window-state terminated
                            transmitted-packets 1
                            received-packets 1
                            minimum 0
                            maximum 0
                            average 0
                            integrity false
                            error-count 0
                            stamp-unrecognized-flag-count 0
                            stamp-malformed-flag-count 0
                            zero-or-negative-delay-count 0
                            duplicate-packet-count 0
                        }
                    }
```

```
                }
            }
        }
    }
```

5.2 Performance monitoring

Performance monitoring encompasses a variety of tools and protocols designed to measure, report, analyze, and optimize network performance, allowing network administrators to detect and resolve issues proactively. This chapter provides information about delay and loss measurement as part of STAMP OAM performance monitoring.

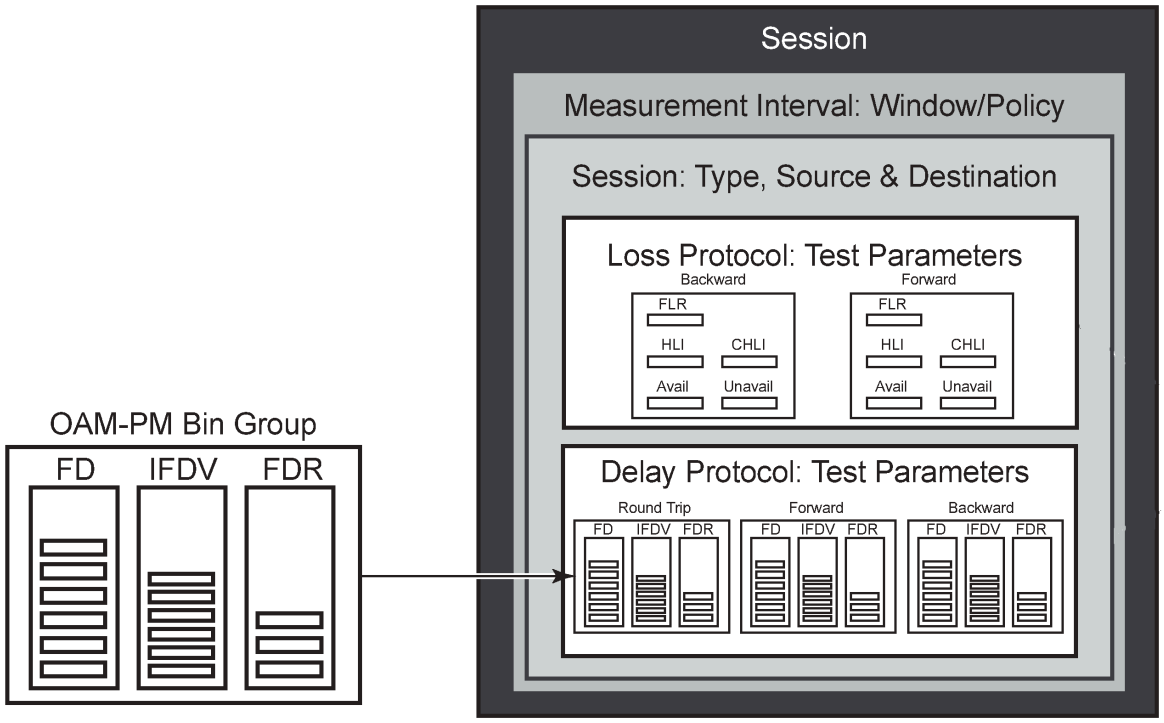
5.2.1 STAMP OAM performance monitoring

The STAMP OAM performance monitoring architecture for gathering and computing Key Performance Indicators (KPIs) using standard protocols and a robust collection model consists of the following foundational components:

- session: This is the overall collection of the test parameters, measurement intervals, thresholds and storage of results. It is the overall container that defines the attributes of the session.
- standard performance monitoring packets: SR Linux supports STAMP to measure performance metrics such as delay and packet loss. See Session sender packet format and Session reflector packet format in the STAMP chapter for more information about STAMP test packets.
- measurement interval: These are time-based non-overlapping windows that capture results that are received in that window of time.
- data structures: These are the unique counters and measurement results that represent the specific protocol.
- bin group: These are ranges in microseconds that count the results that fit into the range.

The following figure shows the hierarchy of the architecture. This figure intends to show the relationship between the components and not to depict all the details of the required parameters.

Figure 5: OAM performance monitoring architecture hierarchy



5.2.1.1 Session

The session is the overall collection of test information fields. The session can be viewed as the single container that combines all aspects of individual tests and the various OAM performance monitoring components. The **session** command includes parameters such as:

- **session-type**: The session type is either proactive or on-demand. The session type setting influences the individual test timing parameters.
- **test-id**: The test identifier is a configured as a numerical value or using the **auto** keyword. An automatically assigned test identifier is released when the test is deleted. Any action that causes the test to be deleted and recreated releases the original test identifier and allocates a new one. These test identifiers are not persistent and not maintained across CPM switchovers. New test identifiers are allocated in the case of CPM switchover.
- test parameters: These include start time, stop time, ability to activate a test, and ability to deactivate a test.
- **measurement-interval**: This is the assignment of collection windows to the session with the appropriate configuration parameters .

Session types and operational states

The operational state of a session is influenced by:

- session type

- session administrative state

In a proactive session, the operational state is up when the administrative state is enabled. The operational state is down when the administrative state is disabled. A proactive test session starts immediately after the **oam performance-monitoring ip session stamp admin-state** command is set to **enable**. A proactive test session stops immediately after the **oam performance-monitoring ip session stamp admin-state** command is set to **disable**. The operational state of a proactive session mirrors the administrative state of the session.

In an on-demand session, the operational state is up when the administrative state is enabled, and the **tools oam performance-monitoring ip session on-demand-action start** command is executed. The on-demand test stops after one of the following actions:

- the completion of the **test-duration** configured using the **oam performance-monitoring ip session session-type on-demand stamp test-duration** command
- the execution of the **tools oam performance-monitoring ip session on-demand-action stop** command

The operational state of an on-demand sessions is not directly tied to the administrative state. It depends on the initiation and completion of tests.

5.2.1.2 Standard performance monitoring packets

SR Linux supports STAMP (Simple Two-way Active Measurement Protocol) to measure performance metrics such as delay and packet loss. STAMP defines test packets in two directions:

- session sender packet. Test request packets that the session sender transmits to the session reflector.
- session reflector packet. Test response packets that the session reflector sends to the session sender.

See the Standard performance monitoring packets and Session reflector packet format in the STAMP chapter for detailed information about STAMP test packets.

5.2.1.3 Data structures

There are two main metrics that are the focus of OAM performance monitoring: delay and loss.

Delay metrics

SR Linux supports the following delay metrics:

- Frame Delay (FD): This measures the time taken for a packet to traverse the network. Any negative FD values are set to zero for binning purposes.
- Frame Delay Range (FDR): This represents the difference between the FD and the lowest FD recorded within a measurement interval. For the first interval, the minimum delay is set to zero. For subsequent intervals, the minimum delay of the previous measurement interval is used as the reference. Negative FD values are set to zero before calculating FDR.
- Inter-Frame Delay Variation (IFDV): Also known as jitter, IFDV measures the variation in delay between adjacent test frames. The absolute difference between the current and previous delay values is calculated, even if the previous delay was negative.

FD, FDR, and IFDV are categorized into bins based on the bin group configuration. The minimum, maximum, and average values for each direction (forward, backward, and round-trip) are also reported.

Delay threshold events and the last time the threshold crossing alarm (TCA) was triggered are also logged .

By default, the average for all delay metrics includes all the results within the measurement interval. However, it is possible to exclude the measurements using **exclude-from-average** for a specified direction. The results are binned but the delay values included in the exclude option are not included in the average computation.

Loss metrics

SR Linux supports the following loss metrics:

- out loss: This metric measures the difference between the packets that the session reflector receives and those that the session sender transmits.
- in loss: This metric calculates the difference between the packets that the session sender receives and those that the session reflector transmits.
- Frame Loss Ratio (FLR): This percentage metric represents the ratio of lost packets during times of availability. FLR is not incremented during periods of unavailability.
- available: The number of delta-ts that are recorded as available. These delta-ts do not exceed the FLR threshold and do not follow an unavailable state. If the available delta-ts follow an unavailable state, they need to fill the availability window before transitioning to available.
- unavailable: The number of delta-ts that are recorded as unavailable. These delta-ts exceed the FLR threshold and do not follow an available state. If the unavailable delta-ts follow and available state, they need to fill the unavailability window before transitioning to unavailable.
- undetermined availability: This counter increments when packets are lost and there is no explicit information about the fate of the packet. This occurs after timeouts and follows an available state.
- undetermined unavailability: Similar to undetermined availability, this counter increments after packet timeouts following an available state when there is no explicit information about the fate of the packet.
- High Loss Interval (HLI): This increments when individual delta-ts reach or exceed the FLR configuration. By default, this is calculated during the availability periods. Executing the **oam performance-monitoring ip session stamp loss hli-force-count** command and configuring the **true** option increments the HLI regardless of the availability state.
- Consecutive High Loss Interval (CHLI): This increments when consecutive HLI intervals meet or exceed a specified threshold within the sliding window. CHLI increments only a single time for each availability window.

5.2.1.4 Measurement intervals

A measurement interval is a window of time that compartmentalizes the gathered measurements for an individual test that has occurred during that time. Allocation of measurement intervals, which equates to system memory, is based on the metrics being collected. This means that when both delay and loss metrics are collected, they allocate their own set of measurement intervals.

Duration

The measurement interval durations are as follows:

- **1-min**
- **5-min**

- **15-min**
- **1-hour**
- **1-day**

Boundary type

The **boundary-type** parameter defines the start of the measurement interval and can be aligned to the local time-of-day clock, with or without an optional offset. By default, the start boundary is **clock-aligned** without an offset. When this configuration is deployed, the measurement interval establishes non-overlapping time-based windows which complete at the specified time. The **boundary-type** parameter can be aligned using the **test-relative** option, which means that the start of the measurement interval coincides with the activation of the test and the length of the measured interval determines the completion.

Clock aligned

When a boundary is **clock-aligned** and **clock-offset** option is configured, a specified amount of time is applied to the measurement interval. Offsets are configured on a per-measurement interval basis and only applicable to clock-aligned measurement intervals. Only offsets less than the measurement interval duration are allowed. The following table lists examples of the start times of each measurement interval.

Table 7: Measurement interval start times

Offset	1-min	5-min	15-min	1-hour	1-day
0 (default)	0, 1, 2, 3, ...	00, 05, 10, 15, ...	00, 15, 30, 45	00 (top of the hour)	midnight
10 minutes	rejected	rejected	10, 25, 40, 55	10 min after the hour	10 min after midnight
30 minutes	rejected	rejected	rejected	30 min after the hour	30 min after midnight
60 minutes	rejected	rejected	rejected	rejected	01:00 AM

Test relative

Although **test-relative** approaches may seem beneficial for simplicity, there are some drawbacks that need to be considered. The goal of the time-based and well-defined collection windows allows for the comparison of measurements across common windows of time throughout the network and for relating different tests or sessions. On-demand tests are typically used for troubleshooting or short term monitoring that does not require alignment or comparison to other performance monitoring data and may make better use of the **test-relative** boundary.

Intervals stored

The statistical data collected and the computed results from each measurement interval are maintained in volatile system memory. The number of intervals stored is configurable per measurement interval. Different measurement intervals have different defaults and ranges. The **interval-stored** parameter defines the number of completed individual test runs to store in volatile memory. There is an additional allocation to account for the active measurement interval. If the retained test data for a measurement interval consumes the final entry, any subsequent entries cause the removal of the oldest data.

Threshold events

The following are the two types of threshold events:

- stateless. The stateless threshold events are:
  - autonomous. Each measurement interval operates independently without carrying forward any information about events from previous intervals.
  - self-contained. The events are evaluated and triggered within the confines of a single measurement interval.
  - enacted when **clear-threshold** is unset.
- stateful. The stateful threshold events are:
  - persistent. The events remain active until specific clear-threshold conditions are met at the end of a subsequent interval.
  - enacted when the clear-threshold is set within the configured range. A value of zero indicates the event clears if no results fall within the specified range in a subsequent interval.

Delay event

Delay events are counter-based events. These are simple counts that compare to the **raise-threshold** for raising and the **clear-threshold** for clearing. These are for each direction, forward, backward, and round-trip. Each of these delay thresholds are raised a maximum of one in a measurement interval when the count in the specified bin or bins reach the **raise-threshold**. The types of delay events are as follows:

- Frame Delay (FD)
- Frame Delay Range (FDR)
- Inter-Frame Delay Variation (IFDV)

Each of these delay threshold events are raised a single time in a measurement interval immediately after the threshold is reached.

Loss events

The types of loss events are as follows:

- counter-based events. These are simple counts that compare to the **raise-threshold** for raising and the **clear-threshold** for clearing. The standard directions, forward and backward as well as a mathematical aggregate that is computed by summing the forward and backward values, are supported. Each of these loss threshold events are raised a maximum of one time in a measurement interval when the count in the specified bin or bins reach the **raise-threshold**.
  - High Loss Interval (HLI) event
  - Consecutive High Loss Interval (CHLI) event
  - unavailability event
  - undetermined availability event
  - undetermined unavailability event
- Average Frame Loss Ratio (Avg-FLR) event. Unlike other loss events, the Avg-FLR event is raised only at the end of the measurement interval. This event does not support aggregated computation. It supports forward and backward directions.

Loss event template

The loss event template is created using the **oam performance-monitoring ip loss loss-events-template** command. All of the loss events are configured using this template. During loss measurement, the loss event template is referenced by a performance monitoring session.

The following considerations apply to loss event templates:

- A loss event template cannot be deleted if it is referenced by a performance monitoring session.
- A loss event template can be modified even if it is referenced by a performance monitoring session without disrupting the ongoing session.
- The reference changes that include, adding a new reference or deleting an existing reference within a loss test session can be done without impacting the performance monitoring session or the loss test session.
- The configuration changes made to loss event templates affect only the loss event function ensuring that performance monitoring continues seamlessly.
- The operational states of the loss events transition based on the configuration changes and the timing of measurement intervals.

5.2.1.5 Bin group

A bin group is a collection of bins where each bin represents a range of delay values. When a delay measurement is performed, the delay results are stored in appropriate bins based on its value. This process is known as binning and it allows for a structured and aggregated view of delay performance over time. Bin groups are created using the **oam performance-monitoring ip delay bin-group** command and referenced by the session using the **oam performance-monitoring ip session stamp delay bin-group** command.

Bin type

There are three types of binnable delay metrics:

- frame delay (FD)
- inter-frame delay variation (IFDV)
- frame delay range (FDR)

All bin types are available in the forward, backward, and round-trip directions. Each of these metrics can have up to ten bin groups configured to group the results.

Bin boundary

Bin groups are configured by indicating a lower boundary. Bin 0 has a lower boundary that is always zero and is not configurable. The microsecond range of the bins is the difference between the adjacent lower boundaries.

For example, **bin-type fd bin 1** configured with **lower-bound 1000** means that:

- bin 0 captures all frame delay statistics results between 0 and 1 ms
- bin 1 captures all results above 1 ms and below the bin 2 lower boundary. Bin 2 is not shown.

The last bin configured represents the bin that collects all the results at and above that lower-bound value. Not all ten bins have to be configured.

Each delay type requires their own values for the bin groups. It is not possible to configure a bin with different values for round trip, forward, and backward. Consider the configuration of the boundaries that represent the important statistics for that specific requirement.

Bin group 1 is the default bin group. Every session requires a bin group to be assigned. By default, bin group 1 is assigned to every performance monitoring session that does not have a bin group explicitly configured. Bin group 1 cannot be modified. Bin group 1 is an automatically created object and not visible in the configuration. If the bin-group 1 is added to the configuration only the mandatory default configuration values may be added. If bin-group 1 is added to the configuration it behaves as non-default bin-groups.

Bin group behaviour

The following considerations apply for bin groups:

- Bin groups cannot be deleted if referenced by a performance monitoring session.
- Bin groups cannot be disabled if referenced by a delay test with the **admin-state** parameter set to *enable*.
- Bin groups can be modified even if referenced by a delay test regardless of the **admin-state** parameter setting. Delay results for the performance monitoring session referencing a changed bin-group is deleted and a new set of bins start recording results.
- Bin group that is excluded from average can be modified even if referenced by a delay test with the **admin-state** parameter set to *enable*.
- Any changes to the attributes of the **oam performance-monitoring ip delay bin-group bin-type delay-event** command do not affect the performance monitoring session or test sessions.

5.2.1.6 Configuring STAMP OAM performance monitoring session

Procedure

To configure a STAMP OAM performance monitoring session, use the **oam performance-monitoring ip session** command and configure the parameters as shown in Configuring a STAMP OAM performance monitoring session.

To allocate a source UDP port to the OAM STAMP application, use the **oam ippm source-udp-port-pools port 64374 application-assignment oam-pm-ip** command and configure the parameters as shown in Allocating source UDP port to OAM performance monitoring application. Configuration of the source UDP port should only be used when explicitly required. If not configured, the source UDP port is dynamically allocated from the dynamic UDP port range by the OAM performance monitoring application.

Example: Configuring a STAMP OAM performance monitoring session

This example configures a STAMP OAM performance monitoring session.

```
--{ + candidate shared default }--[  ]--
# info oam performance-monitoring ip session ip-sess-1
    oam {
        performance-monitoring {
            ip {
                session ip-sess-1 {
                    session-type proactive
                    destination-ip 192.168.1.2
                    destination-udp-port 862
                    source-ip 192.168.1.1
```

```
                    network-instance net-inst-default
                    dscp EF
                    profile in
                    measurement-interval 1-minute {
                        boundary-type clock-aligned
                        clock-offset 0
                        intervals-stored 32
                        threshold-alerts {
                            loss-event enable
                            delay-event enable
                        }
                    }
                    stamp {
                        admin-state enable
                        test-id auto
                        interval 100ms
                        delay {
                            bin-group 2
                        }
                        loss {
                            flr-threshold 5
                            hli-force-count true
                            loss-event loss-1
                            timing {
                                frames-per-delta-t 10
                                consecutive-delta-t 10
                                chli-threshold 5
                            }
                        }
                    }
                }
            }
        }
    }
```

Example: Allocating source UDP port to OAM performance monitoring application

This example allocates a source UDP port to the OAM performance monitoring application.

```
--{ + candidate shared default }--[  ]--
A:srl1# info oam ippm source-udp-port-pools
    oam {
        ippm {
            source-udp-port-pools {
                port 64374 {
                    application-assignment oam-pm-ip
                }
            }
        }
    }
```

5.2.1.7 Performing delay measurement

About this task

Perform the following steps to measure packet delay as part of STAMP OAM performance monitoring:

Procedure

Step 1. Configure a bin group. Use the **oam performance-monitoring ip delay bin-group** command and specify the parameters as shown in the following example.

Example

Configuring a bin group

```
--{ + candidate shared default }--[  ]--
# info oam performance-monitoring ip delay bin-group gp2
    oam {
        performance-monitoring {
            ip {
                delay {
                    bin-group gp2 {
                        admin-state enable
                    }
                }
            }
        }
    }
```

Step 2. Configure the bin type and delay measurement parameters. Use the **oam performance-monitoring ip delay bin-group bin-type** command and specify the **bin-type** and **delay-event** parameters.

Example

Configuring a bin type

```
--{ + candidate shared default }--[  ]--
# info oam performance-monitoring ip delay bin-group 2 bin-type fd
    oam {
        performance-monitoring {
            ip {
                delay {
                    bin-group 2 {
                        bin-type fd {
                            bin 0 {
                                lower-bound 0
                            }
                            bin 1 {
                                lower-bound 500
                            }
                            bin 2 {
                                lower-bound 1000
                            }
                            bin 3 {
                                lower-bound 1500
                            }
                            bin 4 {
                                lower-bound 2000
                            }
                            bin 5 {
                                lower-bound 2500
                            }
                            bin 6 {
                                lower-bound 3000
                            }
                            bin 7 {
                                lower-bound 3500
                            }
                            bin 8 {
```

```
                            lower-bound 4000
                        }
                        bin 9 {
                            lower-bound 4500
                        }
                        delay-event round-trip {
                            lowest-bin 8
                            raise-threshold 10
                            clear-threshold 0
                            exclude-lowest-bin 9
                        }
                    }
                }
            }
        }
    }
}
```

Step 3. Configure the session type. Perform one of the following:

a. To configure a proactive test session, use the **oam performance-monitoring ip session session-type proactive** command.

Example

Configuring a proactive test session

```
--{ + candidate shared default }--[  ]--
# info oam performance-monitoring ip session ip-sess-1 session-type
    oam {
        performance-monitoring {
            ip {
                session ip-sess-1 {
                    session-type proactive
                }
            }
        }
    }
```

b. To configure an on-demand test session, use the **oam performance-monitoring ip session session-type on-demand** command.

Example

Configuring an on-demand test session

```
--{ +* candidate shared default }--[  ]--
# info oam performance-monitoring ip session ip-sess-1 session-type
    oam {
        performance-monitoring {
            ip {
                session ip-sess-1 {
                    session-type on-demand
                }
            }
        }
    }
```

Step 4. Enable STAMP and associate the bin group that has the delay measurement parameters configured. Use the **info oam performance-monitoring ip session stamp** command and specify the STAMP parameters and associate the bin group.

Example

Configuring STAMP parameters for delay measurement

```
--{ +* candidate shared default }--[  ]--
# info oam performance-monitoring ip session ip-sess-1 stamp
    oam {
        performance-monitoring {
            ip {
                session ip-sess-1 {
                    stamp {
                        admin-state enable
                        test-id auto
                        test-duration 60
                        interval 100ms
                        delay {
                            bin-group 2
                        }
                    }
                }
            }
        }
    }
```

Step 5. Start delay measurement. Perform one of the following:

a. When you set the **oam performance-monitoring ip session stamp admin-state** command to **enable**, the proactive test starts immediately.

b. To start an on-demand session, use the **tools oam performance-monitoring ip session on-demand-action start** command.

Expected outcome

The on-demand session starts.

```
--{ + candidate shared default }--[  ]--
# tools oam performance-monitoring ip session ip-sess-1 on-demand-action start
/oam/performance-monitoring/ip/session[session-name=ip-sess-1]:
    OnDemand session started successfully
```

Step 6. Stop delay measurement. Perform one of the following:

a. When you set the **oam performance-monitoring ip session stamp admin-state** command to **disable**, the proactive test stops immediately.

b. To stop an on-demand session, use the **tools oam performance-monitoring ip session on-demand-action stop** command.

Expected outcome

The on-demand session stops.

```
--{ + candidate shared default }--[  ]--
# tools oam performance-monitoring ip session ip-sess-1 on-demand-action stop
/oam/performance-monitoring/ip/session[session-name=ip-sess-1]:
    OnDemand session stopped successfully
```

c. You can use the **oam performance-monitoring ip session stamp test-duration** command to determine the duration of the test and the test stops after the completion of the configured time duration.

Example

Configure time duration for on-demand test

```
--{ +* candidate shared default }--[  ]--
# info oam performance-monitoring ip session ip-sess-1 stamp test-duration
    oam {
        performance-monitoring {
            ip {
                session ip-sess-1 {
                    stamp {
                        test-duration 60
                    }
                }
            }
        }
    }
```

5.2.1.8 Performing loss measurement

About this task

Perform the following steps to measure packet loss as part of STAMP OAM performance monitoring:

Procedure

Step 1. Configure a loss events template. Use the **oam performance-monitoring ip loss loss-events-template** command and specify the loss event parameters as shown in the following example.

Example

Configuring a loss events template

```
--{ +* candidate shared default }--[  ]--
# info oam performance-monitoring ip loss loss-events-template loss-1
    oam {
        performance-monitoring {
            ip {
                loss {
                    loss-events-template loss-1 {
                        hli-event aggregate {
                            raise-threshold 12
                            clear-threshold 6
                        }
                    }
                }
            }
        }
    }
```

Step 2. Configure the session type. Perform one of the following:

a. To configure a proactive test session, use the **oam performance-monitoring ip session session-type proactive** command.

Example

Configuring a proactive test session

```
--{ + candidate shared default }--[  ]--
```

```
# info oam performance-monitoring ip session ip-sess-1 session-type
    oam {
        performance-monitoring {
            ip {
                session ip-sess-1 {
                    session-type proactive
                }
            }
        }
    }
```

b. To configure an on-demand test session, use the **oam performance-monitoring ip session session-type on-demand** command.

Example

Configuring an on-demand test session

```
--{ +* candidate shared default }--[  ]--
# info oam performance-monitoring ip session ip-sess-1 session-type
    oam {
        performance-monitoring {
            ip {
                session ip-sess-1 {
                    session-type on-demand
                }
            }
        }
    }
```

Step 3. Enable STAMP and associate the loss events template that has the loss measurement parameters configured. Use the **oam performance-monitoring ip session stamp** command and specify the parameters as shown in the following example.

Example

Configuring STAMP parameters for loss measurement

```
--{ +* candidate shared default }--[  ]--
# info oam performance-monitoring ip session ip-sess-1 stamp
    oam {
        performance-monitoring {
            ip {
                session ip-sess-1 {
                    stamp {
                        admin-state enable
                        test-id auto
                        test-duration 60
                        interval 100ms
                        delay {
                            bin-group 2
                        }
                        loss {
                            flr-threshold 5
                            hli-force-count true
                            loss-event loss-1
                            timing {
                                frames-per-delta-t 10
                                consecutive-delta-t 10
                                chli-threshold 5
                            }
                        }
                    }
```

```
            }
        }
    }
}
```

Step 4. Start loss measurement. Perform one of the following:

a. When you set the **oam performance-monitoring ip session stamp admin-state** command to **enable**, the proactive test starts immediately.

b. To start an on-demand session, use the **tools oam performance-monitoring ip session on-demand-action start** command.

Expected outcome

The on-demand session starts.

```
--{ + candidate shared default }--[  ]--
# tools oam performance-monitoring ip session session1 on-demand-action start
/oam/performance-monitoring/ip/session[session-name=session1]:
    OnDemand session started successfully
```

Step 5. Stop delay measurement. Perform one of the following:

a. When you set the **oam performance-monitoring ip session stamp admin-state** command to **disable**, the proactive test stops immediately.

b. To stop an on-demand session, use the **tools oam performance-monitoring ip session on-demand-action stop** command.

Expected outcome

The on-demand session stops.

```
--{ + candidate shared default }--[  ]--
# tools oam performance-monitoring ip session session1 on-demand-action stop
/oam/performance-monitoring/ip/session[session-name=session1]:
    OnDemand session stopped successfully
```

c. You can use the **oam performance-monitoring ip session stamp test-duration** command to determine the duration of the test and the test stops after the completion of the configured time duration.

Example

Configure time duration for on-demand test

```
--{ +* candidate shared default }--[  ]--
# info oam performance-monitoring ip session session1 stamp test-duration
    oam {
        performance-monitoring {
            ip {
                session session1 {
                    stamp {
                        test-duration 60
                    }
                }
            }
        }
    }
```

5.2.1.9 Displaying delay and loss measurement results

Procedure

To display the results of delay and loss measurement, use the **info from state oam performance-monitoring ip session** command.

Example: Displaying the results of delay and loss measurement

This example displays the delay and loss measurement results.

```
--{ + candidate shared default }--[ oam performance-monitoring ip ]--
# info from state
        session-type proactive
        destination-ip 192.168.1.2
        destination-udp-port 862
        source-ip 192.168.1.1
        source-udp-port 44000
        network-instance net-inst-default
        dscp EF
        profile in
        ttl 255
        measurement-interval 1-minute {
            boundary-type clock-aligned
            clock-offset 0
            intervals-stored 32
            threshold-alerts {
                loss-event enable
                delay-event enable
            }
        }
        stamp {
            admin-state enable
            oper-state up
            detected-tx-error none
            test-id auto
            test-id-in-use 2147483650
            test-duration 0
            pad-tlv-size 0
            interval 100ms
            statistics {
                stamp-unrecognized-flag-received 0
                stamp-malformed-flag-received 0
            }
            delay {
                bin-group 2
                bin-group-binning active
                measurement-result 1-minute {
                    newest-index 2
                    index 1 {
                        oper-state in-progress
                        suspect-status true
                        start-time 2024-07-05T19:11:54.000Z
                        elapsed-time 6
                        statistics {
                            frames-transmitted 55
                            frames-received 55
                            bin-type fd {
                                forward {
                                    minimum 145
                                    maximum 303
```

```
                    average 208
                }
                backward {
                    minimum 115
                    maximum 241
                    average 170
                }
                round-trip {
                    minimum 290
                    maximum 499
                    average 379
                }
                bin 0 {
                    forward-measurements 55
                    backward-measurements 55
                    round-trip-measurements 55
                }
                bin 1 {
                    forward-measurements 0
                    backward-measurements 0
                    round-trip-measurements 0
                }
                bin 2 {
                    forward-measurements 0
                    backward-measurements 0
                    round-trip-measurements 0
                }
                bin 3 {
                    forward-measurements 0
                    backward-measurements 0
                    round-trip-measurements 0
                }
                bin 4 {
                    forward-measurements 0
                    backward-measurements 0
                    round-trip-measurements 0
                }
                bin 5 {
                    forward-measurements 0
                    backward-measurements 0
                    round-trip-measurements 0
                }
                bin 6 {
                    forward-measurements 0
                    backward-measurements 0
                    round-trip-measurements 0
                }
                bin 7 {
                    forward-measurements 0
                    backward-measurements 0
                    round-trip-measurements 0
                }
                bin 8 {
                    forward-measurements 0
                    backward-measurements 0
                    round-trip-measurements 0
                }
                bin 9 {
                    forward-measurements 0
                    backward-measurements 0
                    round-trip-measurements 0
                }
            }
            bin-type fdr {
```

```
                        forward {
                            minimum 0
                            maximum 157
                            average 51
                        }
                        backward {
                            minimum 0
                            maximum 121
                            average 43
                        }
                        round-trip {
                            minimum 0
                            maximum 181
                            average 68
                        }
                        bin 0 {
                            forward-measurements 55
                            backward-measurements 55
                            round-trip-measurements 55
                        }
                        bin 1 {
                            forward-measurements 0
                            backward-measurements 0
                            round-trip-measurements 0
                        }
                    }
                    bin-type ifdv {
                        forward {
                            minimum 0
                            maximum 158
                            average 43
                        }
                        backward {
                            minimum 0
                            maximum 95
                            average 27
                        }
                        round-trip {
                            minimum 4
                            maximum 130
                            average 54
                        }
                        bin 0 {
                            forward-measurements 50
                            backward-measurements 54
                            round-trip-measurements 44
                        }
                        bin 1 {
                            forward-measurements 4
                            backward-measurements 0
                            round-trip-measurements 10
                        }
                        bin 2 {
                            forward-measurements 0
                            backward-measurements 0
                            round-trip-measurements 0
                        }
                        bin 3 {
                            forward-measurements 0
                            backward-measurements 0
                            round-trip-measurements 0
                        }
                        bin 4 {
                            forward-measurements 0
```

```
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 5 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 6 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 7 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 8 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 9 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                }
            }
        }
        index 2 {
            oper-state in-progress
            suspect-status false
            start-time 2024-07-05T19:12:00.000Z
            elapsed-time 3
            statistics {
                frames-transmitted 32
                frames-received 32
                bin-type fd {
                    forward {
                        minimum 158
                        maximum 268
                        average 219
                    }
                    backward {
                        minimum 98
                        maximum 281
                        average 172
                    }
                    round-trip {
                        minimum 294
                        maximum 512
                        average 391
                    }
                    bin 0 {
                        forward-measurements 32
                        backward-measurements 32
                        round-trip-measurements 30
                    }
                    bin 1 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 2
```

```
                    }
                    bin 2 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 3 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 4 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 5 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 6 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 7 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 8 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 9 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                }
                bin-type fdr {
                    forward {
                        minimum 13
                        maximum 123
                        average 74
                    }
                    backward {
                        minimum 0
                        maximum 183
                        average 72
                    }
                    round-trip {
                        minimum 4
                        maximum 222
                        average 101
                    }
                    bin 0 {
                        forward-measurements 32
                        backward-measurements 32
                        round-trip-measurements 32
                    }
                    bin 1 {
```

```
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                }
                bin-type ifdv {
                    forward {
                        minimum 1
                        maximum 97
                        average 24
                    }
                    backward {
                        minimum 0
                        maximum 109
                        average 51
                    }
                    round-trip {
                        minimum 4
                        maximum 174
                        average 62
                    }
                    bin 0 {
                        forward-measurements 32
                        backward-measurements 29
                        round-trip-measurements 28
                    }
                    bin 1 {
                        forward-measurements 0
                        backward-measurements 3
                        round-trip-measurements 4
                    }
                    bin 2 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 3 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 4 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 5 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 6 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 7 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 8 {
                        forward-measurements 0
                        backward-measurements 0
```

```
                        round-trip-measurements 0
                    }
                    bin 9 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                }
            }
        }
    }
    measurement-result raw {
        newest-index 1
        index 1 {
            oper-state in-progress
            suspect-status true
            start-time 2024-07-05T19:11:54.000Z
            elapsed-time 9
            statistics {
                frames-transmitted 87
                frames-received 87
                bin-type fd {
                    forward {
                        minimum 145
                        maximum 303
                        average 212
                    }
                    backward {
                        minimum 98
                        maximum 281
                        average 171
                    }
                    round-trip {
                        minimum 290
                        maximum 512
                        average 383
                    }
                    bin 0 {
                        forward-measurements 87
                        backward-measurements 87
                        round-trip-measurements 85
                    }
                    bin 1 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 2
                    }
                    bin 2 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 3 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 4 {
                        forward-measurements 0
                        backward-measurements 0
                        round-trip-measurements 0
                    }
                    bin 5 {
                        forward-measurements 0
```

```
                backward-measurements 0
                round-trip-measurements 0
            }
            bin 6 {
                forward-measurements 0
                backward-measurements 0
                round-trip-measurements 0
            }
            bin 7 {
                forward-measurements 0
                backward-measurements 0
                round-trip-measurements 0
            }
            bin 8 {
                forward-measurements 0
                backward-measurements 0
                round-trip-measurements 0
            }
            bin 9 {
                forward-measurements 0
                backward-measurements 0
                round-trip-measurements 0
            }
        }
        bin-type fdr {
            forward {
                minimum 0
                maximum 157
                average 60
            }
            backward {
                minimum 0
                maximum 183
                average 53
            }
            round-trip {
                minimum 0
                maximum 222
                average 80
            }
            bin 0 {
                forward-measurements 87
                backward-measurements 87
                round-trip-measurements 87
            }
            bin 1 {
                forward-measurements 0
                backward-measurements 0
                round-trip-measurements 0
            }
        }
        bin-type ifdv {
            forward {
                minimum 0
                maximum 158
                average 36
            }
            backward {
                minimum 0
                maximum 109
                average 36
            }
            round-trip {
                minimum 4
```

```
                            maximum 174
                            average 57
                        }
                        bin 0 {
                            forward-measurements 82
                            backward-measurements 83
                            round-trip-measurements 72
                        }
                        bin 1 {
                            forward-measurements 4
                            backward-measurements 3
                            round-trip-measurements 14
                        }
                        bin 2 {
                            forward-measurements 0
                            backward-measurements 0
                            round-trip-measurements 0
                        }
                        bin 3 {
                            forward-measurements 0
                            backward-measurements 0
                            round-trip-measurements 0
                        }
                        bin 4 {
                            forward-measurements 0
                            backward-measurements 0
                            round-trip-measurements 0
                        }
                        bin 5 {
                            forward-measurements 0
                            backward-measurements 0
                            round-trip-measurements 0
                        }
                        bin 6 {
                            forward-measurements 0
                            backward-measurements 0
                            round-trip-measurements 0
                        }
                        bin 7 {
                            forward-measurements 0
                            backward-measurements 0
                            round-trip-measurements 0
                        }
                        bin 8 {
                            forward-measurements 0
                            backward-measurements 0
                            round-trip-measurements 0
                        }
                        bin 9 {
                            forward-measurements 0
                            backward-measurements 0
                            round-trip-measurements 0
                        }
                    }
                }
            }
        }
        delay-events fd direction round-trip {
            last-tca-time 1970-01-01T00:00:00.000Z
            oper-state not-active
        }
        delay-events ifdv direction round-trip {
            last-tca-time 1970-01-01T00:00:00.000Z
            oper-state not-active
```

```
            }
        }
        loss {
            flr-threshold 5
            hli-force-count true
            loss-event loss-1
            timing {
                frames-per-delta-t 10
                consecutive-delta-t 10
                chli-threshold 5
            }
            measurement-result 1-minute {
                newest-index 2
                index 1 {
                    oper-state in-progress
                    suspect-status true
                    start-time 2024-07-05T19:11:54.000Z
                    elapsed-time 6
                    statistics {
                        frames-transmitted 55
                        frames-received 55
                        forward {
                            out-loss 0
                            available 0
                            unavailable 0
                            undetermined-available 0
                            undetermined-unavailable 0
                            high-loss-intervals 0
                            consecutive-high-loss-intervals 0
                            minimum-frame-loss-ratio 0
                            maximum-frame-loss-ratio 0
                            average-frame-loss-ratio 0
                        }
                        backward {
                            in-loss 0
                            available 0
                            unavailable 0
                            undetermined-available 0
                            undetermined-unavailable 0
                            high-loss-intervals 0
                            consecutive-high-loss-intervals 0
                            minimum-frame-loss-ratio 0
                            maximum-frame-loss-ratio 0
                            average-frame-loss-ratio 0
                        }
                    }
                }
                index 2 {
                    oper-state in-progress
                    suspect-status false
                    start-time 2024-07-05T19:12:00.000Z
                    elapsed-time 3
                    statistics {
                        frames-transmitted 32
                        frames-received 32
                        forward {
                            out-loss 0
                            available 0
                            unavailable 0
                            undetermined-available 0
                            undetermined-unavailable 0
                            high-loss-intervals 0
                            consecutive-high-loss-intervals 0
                            minimum-frame-loss-ratio 0
```

```
                            maximum-frame-loss-ratio 0
                            average-frame-loss-ratio 0
                        }
                        backward {
                            in-loss 0
                            available 0
                            unavailable 0
                            undetermined-available 0
                            undetermined-unavailable 0
                            high-loss-intervals 0
                            consecutive-high-loss-intervals 0
                            minimum-frame-loss-ratio 0
                            maximum-frame-loss-ratio 0
                            average-frame-loss-ratio 0
                        }
                    }
                }
            }
            measurement-result raw {
                newest-index 1
                index 1 {
                    oper-state in-progress
                    suspect-status true
                    start-time 2024-07-05T19:11:54.000Z
                    elapsed-time 9
                    statistics {
                        frames-transmitted 87
                        frames-received 87
                        forward {
                            out-loss 0
                            available 0
                            unavailable 0
                            undetermined-available 0
                            undetermined-unavailable 0
                            high-loss-intervals 0
                            consecutive-high-loss-intervals 0
                            minimum-frame-loss-ratio 0
                            maximum-frame-loss-ratio 0
                            average-frame-loss-ratio 0
                        }
                        backward {
                            in-loss 0
                            available 0
                            unavailable 0
                            undetermined-available 0
                            undetermined-unavailable 0
                            high-loss-intervals 0
                            consecutive-high-loss-intervals 0
                            minimum-frame-loss-ratio 0
                            maximum-frame-loss-ratio 0
                            average-frame-loss-ratio 0
                        }
                    }
                }
            }
            loss-events hli direction aggregate {
                last-tca-time 1970-01-01T00:00:00.000Z
                oper-state not-active
            }
        }
    }
}
```

6 sFlow

sFlow is used to monitor data traffic flows traversing different points in a network. The sFlow functionality uses an sFlow agent and an sFlow collector. The agent is software that runs on a network element and samples and reports flow headers and statistics. The collector is software that typically runs on a remote server and receives the flow headers and statistics from one or more sFlow agents.

Sampling and reporting are accomplished as the sFlow agent running on a network element takes periodic samples of ingress traffic and reports the data to one or more collectors. The network element does not need to maintain a local flow cache. Instead, the sampled header information is immediately sent to the collector without additional processing.

SR Linux supports sFlow version 5 behavior and formats. On 7250 IXR chassis-based systems, sFlow is implemented in hardware. On 7220 IXR systems, sFlow functionality is implemented in software. sFlow behavior is identical on both platforms, with the following exceptions:

- DSCP configuration is supported only on 7250 IXR systems.
- Frame sample sizes of 256 or 512 bytes are supported only on 7250 IXR systems.
- IPv6 sFlow collector configuration is supported only on 7250 IXR systems.

6.1 sFlow sampling

sFlow works by sampling flow data and reporting the samples to the configured sFlow collectors. Based on the configured system sampling rate, the forwarding plane samples ingress packet flows and sends the sampled headers to the sFlow agent in the control plane.

All ingress packets are subject to sampling. By default, 256 bytes are sampled from each packet. Each sample includes the following:

- 7220 IXR systems – samples include the top 256 bytes of the sampled packet, starting at the outer Ethernet header
- 7250 IXR systems – samples include the top 256 or 512 bytes of the sampled packet, starting at the outer Ethernet header

The sampled packets are sent to the configured sFlow collectors with the sampled data in sFlow raw packet data format.

For sampled IPv4 packets, the IPv4 header data fields are sent with the raw data. For sampled IPv6 packets, the IPv6 header data fields are sent with the raw data.

6.2 sFlow collector reporting

sFlow reports sampled headers and statistics to the configured collectors using IP/UDP datagrams. UDP port 6343 is the default destination port, but you can optionally configure a different port. Sampled packets are sent as soon as the samples are taken, and interface statistics are sent at 10 second intervals. SR Linux supports up to eight remote IPv4 sFlow collectors or one remote IPv6 sFlow collector. IPv6 sFlow collector configuration is supported only on 7250 IXR systems. IPv4 and IPv6 sFlow collectors are

mutually exclusive and cannot be configured simultaneously. Each collector can only have one IPv4 or IPv6 address. The flow and counter samples are aggregated in an sFlow datagram packet in software implementation.

sFlow DSCP settings

On 7220 IXR systems, flow and counter samples are assigned a non-configurable default DSCP value of 0.

On 7250 IXR systems:

- Flow samples are also assigned a default DSCP value of 0, but you can optionally assign a different DSCP value for flow samples that applies to all collectors.
- Counter samples are assigned a default DSCP value of 34, which cannot be modified.

6.3 sFlow counter samples

Another aspect of the sFlow agent is streaming of interface statistics to configured sFlow collectors. Statistics are only sent to a collector if sFlow has been enabled on an interface. Interface statistics are sent based on a default poll-interval of 10 seconds with a separate timer for each interface. When the interval expires. the current value of each associated statistics are sent to the configured collectors.

The interface counter sample contains:

- Interface index
- Interface type
- Interface speed
- Oper and admin status
- Input octets
- Input packets
- Input broadcast packets
- Input discards packets
- Output errors
- Output octets
- Output packets
- Output broadcast packets
- Output discards packets

6.4 Configuring the sFlow agent

Procedure

To configure the sFlow agent on the system, you enable sFlow, and optionally configure the sampling rate (by default, 1 out of every 10 000 packets) and sample size (by default, 256 bytes are sampled from each packet).

Example: Configuring the sFlow agent

The following example enables sFlow on the system and configures the system sampling rate and sample size. The polling interval is not configurable. The following sample size options apply:

-
- 7220 IXR-D2, D3, D4, D5, and 7220 IXR-H systems: 256 bytes
- 7250 IXR 6/10/6e/10e/ and 7250 IXR-X3b systems: 256 or 512 bytes

```
--{ * candidate shared default }--[  ]--
# info
    system {
        sflow {
            admin-state enable
            sample-rate 50000
            sample-size 512
        }
    }
```

6.5 Configuring sFlow collectors

Procedure

The sFlow agent sends sampled packets to sFlow collectors. You can configure up to eight IPv4 sFlow collectors or one IPv6 sFlow collector to receive the data. IPv6 sFlow collector configuration is supported only on 7250 IXR systems. IPv4 and IPv6 sFlow collectors are mutually exclusive and cannot be configured simultaneously. To configure an sFlow collector, you specify its IP address, associated network instance, and IP address to be used as the source IP address in sFlow packets sent from SR Linux to the collector. You can optionally specify a destination port (by default, this is UDP port 6343).

Note:
Configuring a network-instance is mandatory. Also, a collector cannot be reached using the **mgmt** network-instance.

Example: Configuring IPv4 sFlow collectors

The following example configures two IPv4 sFlow collectors. The IP address for each collector is configured, as well as its network instance and source IP address. Each collector receives all samples. The collector DSCP value for flow samples is also configured (applicable only on 7250 IXR systems). If no value is specified, the default DSCP value of 0 applies.

```
--{ * candidate shared }--[  ]--
#info system sflow
    system {
        sflow {
            dscp 14
            collector 1 {
                collector-address 10.50.4.1
                source-address 192.0.2.1
                network-instance default
            }
            collector 2 {
                collector-address 10.50.4.2
                source-address 10.1.5.2
                network-instance default
```

```
                port 4310
            }
        }
    }
```

Example: Configuring an IPv6 sFlow collector

The following example configures one IPv6 sFlow collector. The IP address for the collector is configured, as well as its network instance and source IP address. The collector receives all samples.

Note: Only one IPv6 collector with a **collector** value of 1 can be configured.

```
--{ * candidate shared default }--[  ]--
# info system sflow
    system {
        sflow {
            collector 1 {
                collector-address 2001:db8::1
                network-instance default
                source-address 2001:db8::2
            }
        }
    }
```

6.6 Configuring sFlow for an interface

Procedure

When sFlow is configured for an Ethernet or a LAG interface, the ingress packets are taken for sampling according to the **sample-rate**.

The following considerations apply for sFlow on a LAG interface:

- sFlow on LAG feature is available on 7250 IXR-6/10/6e/10e and 7250 IXR-X3b platforms.
- When sFlow on LAG interface is disabled, the sFlow state of the member ports are also disabled.
- When sFlow on LAG interface is enabled, the sFlow state of the member ports follow the individual sFlow admin state that is configured. The default value is **enable**.
- The **Input interface** field in the flow samples of the ingress traffic collected on the LAG port displays the **ifIndex** of the LAG member port.
- The **Output interface** field in the flow samples of the ingress unicast traffic that is marked as egress via a LAG port displays the **ifIndex** of the LAG port.

Example: Configuring sFlow for an Ethernet interface

The following example enables sFlow on an Ethernet interface.

```
--{ * candidate shared default }--[  ]--
# info interface ethernet-1/1
    interface ethernet-1/1 {
        admin-state enable
        sflow {
            admin-state enable
        }
```

```
    }
```

Example: Configuring sFlow for a LAG interface

The following example enables sFlow on a LAG interface.

```
--{ + candidate shared default }--[  ]--
# info interface lag1
    interface lag1 {
        sflow {
            admin-state enable
        }
        lag {
            lag-type static
            min-links 2
        }
    }
```

6.7 Displaying the state of the sFlow agent

Procedure

To display the system-wide state of the sFlow agent, including any sFlow parameters, collector configuration, and general statistics, use the **info from state** command in candidate or running mode, or the **info** command in state mode.

Example: Info from state command

```
# info from state system sflow
    system {
        sflow {
            admin-state enable
            sample-rate 1000
            sample-size 256
            collector 1 {
                collector-address 10.1.1.24
                network-instance default
                source-address 10.0.0.1
                port 6343
                next-hop 172.24.71.65
            }
            statistics {
                total-samples-taken 5457
                total-sent-packets 26800
            }
        }
    }
```

6.8 Displaying the status of the sFlow agent

Procedure

Use the **show system sflow status** command in show mode to display the general status of the sFlow agent:

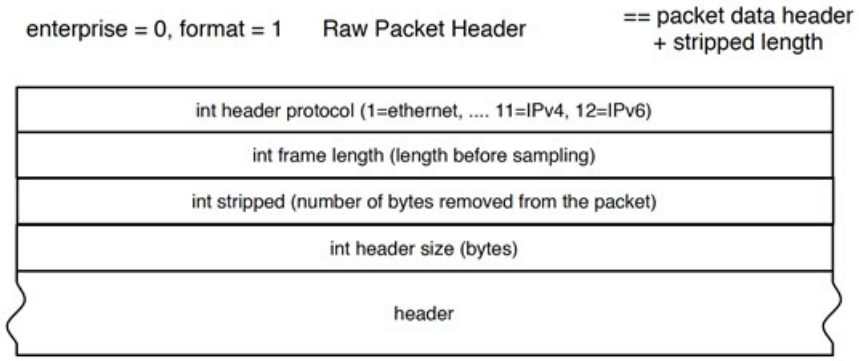
Example: Show system sflow status command

```
--{ running }--[  ]--
# enter show
# show system sflow status
-------------------------------------
Admin State            : enable
Sample Rate            : 10000
Sample Size            : 256
DSCP                   : 0
Total Samples          : 0
Total Collector Packets: 3269158
-------------------------------------
collector-id     : 8
collector-address: 172.10.10.10
network-instance : default
source-address   : 10.0.0.1
port             : 6343
next-hop         : 172.24.71.65
-------------------------------------
```

6.9 sFlow formats

Figure 6: Raw packet header shows an example of a raw packet header for an sFlow format.

Figure 6: Raw packet header



6.10 Sampled data and counter examples

The following is an example of IPv4 flow sample data:

Example: IPv4 flow sample data

```
InMon sFlow
    Datagram version: 5
    Agent address type: IPv4 (1)
    Agent address: 10.0.0.1
```

```
    Sub-agent ID: 2
    Sequence number: 0
    SysUptime: 0
    NumSamples: 1
    Flow sample, seq 0
        0000 0000 0000 0000 0000 .... .... .... = Enterprise: standard sFlow (0)
        .... .... .... .... .... 0000 0000 0001 = sFlow sample type: Flow sample (1)
        Sample length (byte): 141
        Sequence number: 0
        0000 0000 .... .... .... .... .... .... = Source ID class: 0
        .... .... 0000 0000 0000 0000 0011 0110 = Index: 54
        Sampling rate: 1 out of 5 packets
        Sample pool: 0 total packets
        Dropped packets: 0
        Input interface (ifIndex): 54
        .000 0000 0000 0000 0000 0000 0011 0110 = Output interface (ifIndex): 54
        Flow record: 1
        Raw packet header
            0000 0000 0000 0000 0000 .... .... .... = Enterprise: standard sFlow (0)
            Format: Raw packet header (1)
            Flow data length (byte): 101
            Header protocol: Ethernet (1)
            Frame Length: 98
            Payload removed: 0
            Original packet length: 85
            Header of sampled packet:
                000c0002000000000011111108004500005200000000004006...
                Ethernet II, Src: 00:00:00_11:11:11 (00:00:00:11:11:11),
                    Dst: BebIndus_02:00:00 (00:0c:00:02:00:00)
                    Destination: BebIndus_02:00:00 (00:0c:00:02:00:00)
                    Source: 00:00:00_11:11:11 (00:00:00:11:11:11)
                    Type: IPv4 (0x0800)
                Internet Protocol Version 4, Src: 10.100.1.2, Dst: 10.1.1.2
                    0100 .... = Version: 4
                    .... 0101 = Header Length: 20 bytes (5)
                    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
                    Total Length: 82
                    Identification: 0x0000 (0)
                    Flags: 0x00
                    Fragment offset: 0
                    Time to live: 64
                    Protocol: TCP (6)
                    Header checksum: 0x35a1 [validation disabled]
                    [Header checksum status: Unverified]
                    Source: 10.100.1.2
                    Destination: 10.1.1.254
                    [Source GeoIP: Unknown]
                    [Destination GeoIP: Unknown]
                Transmission Control Protocol, Src Port: 0, Dst Port: 0, Seq: 0
                LBT-TCP Protocol
                LBMC Protocol
                [Unreassembled Packet: LBT-TCP]
```

The following is an example of IPv6 flow sample data:

Example: IPv6 flow sample data

```
InMon sFlow
    Datagram version: 5
    Agent address: 3000::2 (3000::2)
    Sub-agent ID: 24
    Sequence number: 1011
    SysUptime: 63684188
```

```
    NumSamples: 1
    Flow sample, seq 2368
        Enterprise: standard sFlow (0)
        sFlow sample type: Flow sample (1)
        Sample length (byte): 568
        Sequence number: 2368
        Source ID class: 0 index: 704510
        Sampling rate: 1 out of 1 packets
        Sample pool: 0 total packets
        Dropped packets: 0
        Input interface: ifIndex 134922238
        Output interface: ifIndex 0
        Flow record: 1
        Raw packet header
            Enterprise: standard sFlow (0)
            Format: Raw packet header (1)
            Flow data length (byte): 528
            Header protocol: Ethernet (1)
            Frame Length: 125 bytes
            Payload removed: 0 bytes
            Header of sampled packet: 01005e000002000103ff02018100c064080045c0006b3005...
                Ethernet II, Src: 3com_ff:02:01 (00:01:03:ff:02:01), Dst:
                IPv4mcast_00:00:02 (01:00:5e:00:00:02)
                802.1Q Virtual LAN, PRI: 6, CFI: 0, ID: 100
                Internet Protocol Version 4, Src: 192.35.1.1 (192.35.1.1),
                Dst: 224.0.0.2 (224.0.0.2)
                User Datagram Protocol, Src Port: ldp (646), Dst Port: ldp (646)
                Label Distribution Protocol
                    Version: 1
                    PDU Length: 75
                    LSR ID: 3.3.3.1 (3.3.3.1)
                    Label Space ID: 0
                    Hello Message
```

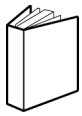
The following is a counter sample example:

Example: Counters sample

```
InMon sFlow
    Datagram version: 5
    Agent address: 10.0.0.1 (10.0.0.1)
    Sub-agent ID: 0
    Sequence number: 8
    SysUptime: 6548000
    NumSamples: 1
    Counters sample, seq 1
        Enterprise: standard sFlow (0)
        sFlow sample type: Counters sample (2)
        Sample length (byte): 108
        Sequence number: 1
        Source ID type: 64
        Source ID index: 49150
        Counters records: 1
        Generic interface counters
            Enterprise: standard sFlow (0)
            Format: Generic interface counters (1)
            Flow data length (byte): 88
            Interface index: 1073790974
            Interface Type: 6
            Interface Speed: 25600
            IfDirection: Full-Duplex
            IfAdminStatus: Up
            IfOperStatus: Up
```

```
            Input Octets: 0
            Input Packets: 0
            Input Multicast Packets: 0
            Input Broadcast Packets: 0
            Input Discarded Packets: 0
            Input Errors: 0
            Input Unknown Protocol
        Packets: 0
            Output Octets: 0
            Output Packets: 0
            Output Multicast Packets: 0
            Output Broadcast Packets: 0
            Output Discarded Packets: 0
            Output Errors: 0
            Promiscuous Mode: 0
```

Customer document and product support



Customer documentation

Customer documentation welcome page

Technical support

Product support portal

Documentation feedback

Customer documentation feedback