

Internet-Draft
Network Working Group
Intended Status: Informational
Expires: January 18, 2016

Bhuvaneswaran Vengainathan
Anton Basil
Veryx Technologies
Mark Tassinari
Hewlett-Packard
Vishwas Manral
Ionos Corp
Sarah Banks
VSS Monitoring
July 19, 2015

Benchmarking Methodology for SDN Controller Performance
draft-bhuvan-bmwg-sdn-controller-benchmark-meth-01

Abstract

This document defines the methodologies for benchmarking performance of SDN controllers. Terminology related to benchmarking SDN controllers is described in the companion terminology document. SDN controllers have been implemented with many varying designs in order to achieve their intended network functionality. Hence, the authors have taken the approach of considering an SDN controller as a black box, defining the methodology in a manner that is agnostic to protocols and network services supported by controllers. The intent of this document is to provide a standard mechanism to measure the performance of all controller implementations.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 18, 2016.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Scope	3
3. Test Setup	3
3.1 Test setup - Controller working in Standalone Mode	4
3.2 Test setup - Controller working in Cluster Mode	5
4. Test Considerations	6
4.1 Network Topology	6
4.2 Test Traffic	6
4.3 Connection Setup	6
4.4 Measurement Point Specification and Recommendation	7
4.5 Connectivity Recommendation	7
4.6 Test Repeatability	7
5. Test Reporting	7
6. Benchmarking Tests	8
6.1 Performance	8
6.1.1 Network Topology Discovery Time	8
6.1.2 Asynchronous Message Processing Time	9
6.1.3 Asynchronous Message Processing Rate	11
6.1.4 Reactive Path Provisioning Time	12
6.1.5 Proactive Path Provisioning Time	13
6.1.6 Reactive Path Provisioning Rate	14
6.1.7 Proactive Path Provisioning Rate	15
6.1.8 Network Topology Change Detection Time	16
6.2 Scalability	17
6.2.1 Control Sessions Capacity	17
6.2.2 Network Discovery Size	18
6.2.3 Forwarding Table Capacity	19
6.3 Security	20
6.3.1 Exception Handling	20
6.3.2 Denial of Service Handling	21
6.4 Reliability	22
6.4.1 Controller Failover Time	22
6.4.2 Network Re-Provisioning Time	23

7. References	24
7.1 Normative References	24
7.2 Informative References	25
8. IANA Considerations	25
9. Security Considerations	25
10. Appendix A - Example Test Topologies	26
11. Appendix B - Benchmarking Methodology using OF Controllers	26
12. Acknowledgements	45
13. Authors' Addresses	46

1. Introduction

This document provides generic methodologies for benchmarking SDN controller performance. An SDN controller may support many northbound and southbound protocols, implement a wide range of applications, and work solely, or as a group to achieve the desired functionality. This document considers an SDN controller as a black box, regardless of design and implementation. The tests defined in the document can be used to benchmark SDN controller for performance, scalability, reliability and security independent of northbound and southbound protocols. These tests can be performed on an SDN controller running as a virtual machine (VM) instance or on a bare metal server. This document is intended for those who want to measure the SDN controller performance as well as compare various SDN controllers performance.

Conventions used in this document

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

2. Scope

This document defines methodology to measure the networking metrics of SDN controllers. The tests defined in this document enable benchmarking of SDN Controllers in two ways; as a standalone controller and as a cluster of homogeneous controllers. These tests are recommended for execution in lab environments rather than in live network deployments. Performance benchmarking of a federation of controllers is beyond the scope of this document.

3. Test Setup

The tests defined in this document enable measurement of an SDN controllers performance in standalone mode and cluster mode. This section defines common reference topologies that are later referred to in individual tests.

3.1 Test setup - Controller working in Standalone Mode

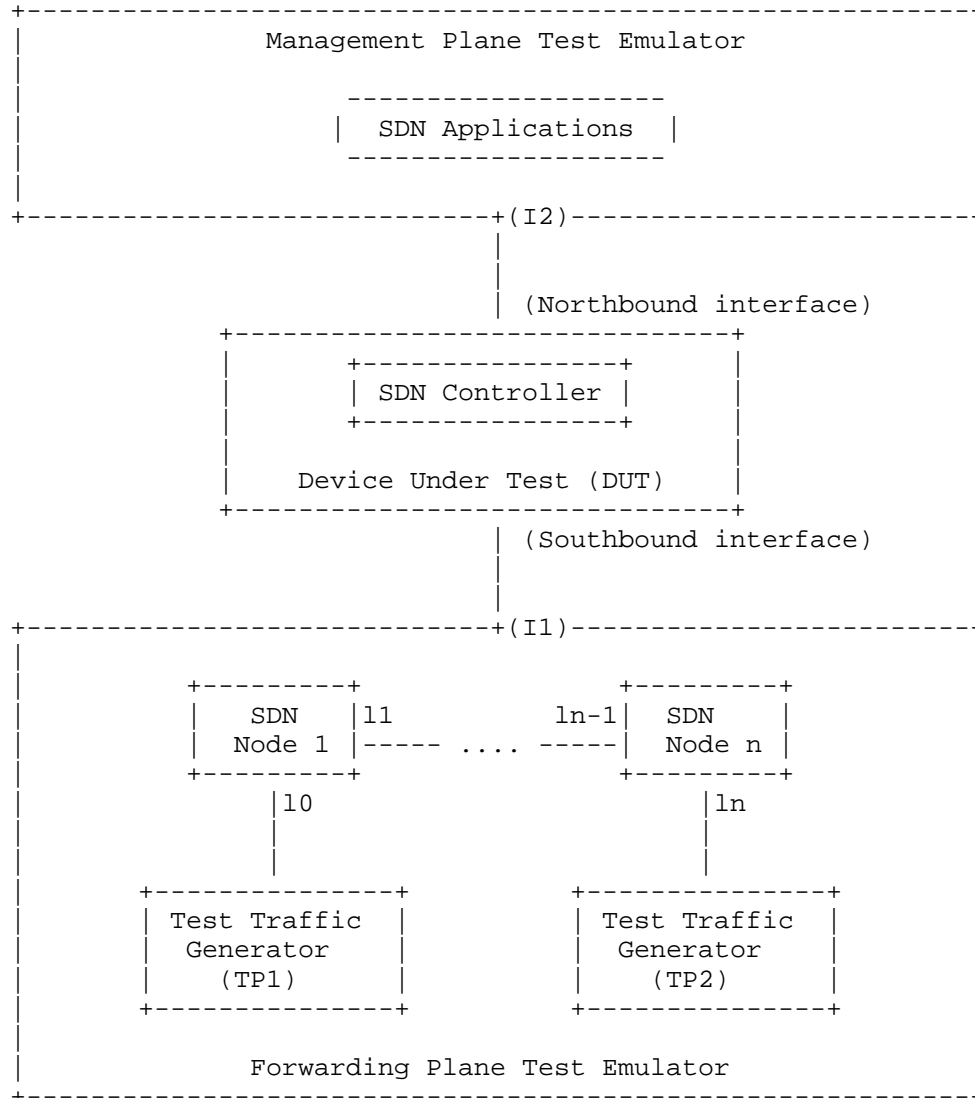


Figure 1

3.2 Test setup - Controller working in Cluster Mode

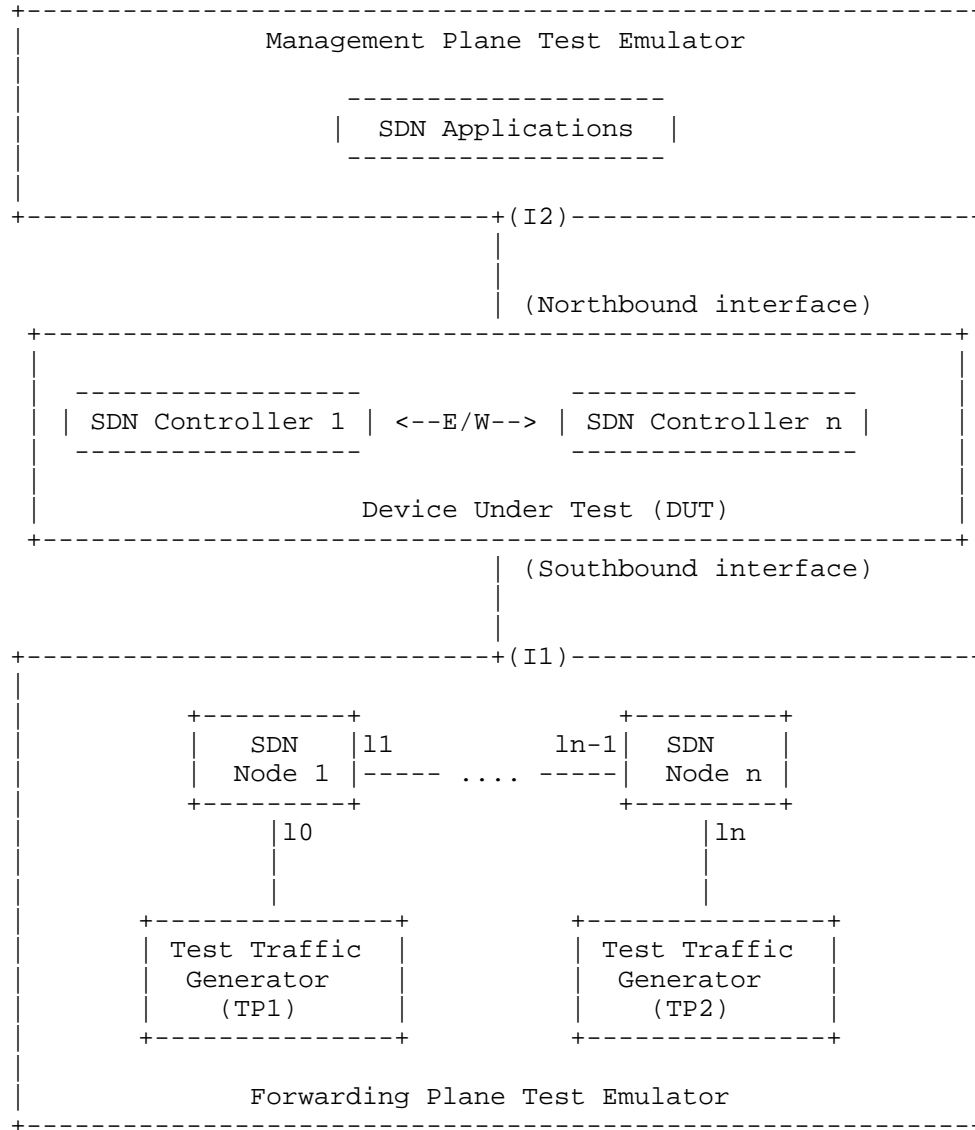


Figure 2

4. Test Considerations

4.1 Network Topology

The test cases SHOULD use Leaf-Spine topology with atleast 1 SDN node in the topology for benchmarking. The test traffic generators TP1 and TP2 SHOULD be connected to the first and the last SDN leaf node. If a test case uses test topology with 1 SDN node, the test traffic generators TP1 and TP2 SHOULD be connected to the same node. However to achieve a complete performance characterization of the SDN controller, it is recommended that the controller be benchmarked for many network topologies and varying number of SDN nodes. This document includes a few sample test topologies, defined in Section 10 - Appendix A for reference. Further, care should be taken to make sure that a loop prevention mechanism is enabled either in the SDN controller, or in the network when the topology contains redundant network paths.

4.2 Test Traffic

Test traffic are used to notify the controller about the arrival of new flows. The test cases SHOULD use multiple frame sizes as recommended in RFC 2544 for benchmarking.

4.3 Connection Setup

There may be controller implementations that support unencrypted and encrypted network connections with SDN nodes. Further, the controller may have backward compatibility with SDN nodes running older versions of southbound protocols. It is recommended that the controller performance be measured with one or more applicable connection setup methods defined below.

1. Unencrypted connection with SDN nodes, running same protocol version.
2. Unencrypted connection with SDN nodes, running different protocol versions.

Example:

1. Controller running current protocol version and switch running older protocol version
2. Controller running older protocol version and switch running current protocol version
3. Encrypted connection with SDN nodes, running same protocol version

4. Encrypted connection with SDN nodes, running different protocol versions.

Example:

1. Controller running current protocol version and switch running older protocol version
2. Controller running older protocol version and switch running current protocol version

4.4 Measurement Point Specification and Recommendation

The measurement accuracy depends on several factors including the point of observation where the indications are captured. For example, the notification can be observed at the controller or test emulator. The test operator SHOULD make the observations/measurements at the interfaces of test emulator unless it is explicitly mentioned otherwise in the individual test.

4.5 Connectivity Recommendation

The SDN controller in the test setup SHOULD be connected directly with the forwarding and the management plane test emulators to avoid any delays or failure introduced by the intermediate devices during benchmarking tests.

4.6 Test Repeatability

To increase the confidence in measured result, it is recommended that this test SHOULD be performed atleast 10 times with same number of nodes using same topology.

5. Test Reporting

Each test has a reporting format which is specific to individual tests. In addition, the following test configuration parameters and controller settings parameters MUST be reflected in the test report.

Test Configuration Parameters:

1. Controller name and version
2. Northbound protocols and versions
3. Southbound protocols and versions
4. Controller redundancy mode (Standalone or Cluster Mode)
5. Connection setup (Unencrypted or Encrypted)
6. Network Topology (Mesh or Tree or Linear)

7. SDN Node Type (Physical or Virtual or Emulated)
8. Number of Nodes
9. Number of Links
10. Test Traffic Type
11. Controller System Configuration (e.g., CPU, Memory, Operating System, Interface Speed etc.,)
12. Reference Test Setup (e.g., Section 3.1 etc.,)

Controller Settings Parameters:

1. Topology re-discovery timeout
2. Controller redundancy mode (e.g., active-standby etc.,)

6. Benchmarking Tests

6.1 Performance

6.1.1 Network Topology Discovery Time

Objective:

Measure the time taken by the SDN controller to discover the network topology (nodes and links), expressed in milliseconds.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document.

Prerequisite:

1. The controller MUST support network discovery.
2. Tester should be able to retrieve the discovered topology information either through the controller's management interface or northbound interface to determine if the discovery was successful and complete.
3. Ensure that the controller's topology re-discovery timeout has been set to the maximum value to avoid initiation of re-discovery process in the middle of the test.

Procedure:

1. Ensure that the controller is operational, its network applications, northbound and southbound interfaces are up and running.
2. Establish the network connections between controller and SDN nodes.
3. Record the time for the first discovery message (Tm1) received from the controller at forwarding plane test emulator interface I1.

4. Query the controller every 3 seconds to obtain the discovered network topology information through the northbound interface or the management interface and compare it with the deployed network topology information.
5. Stop the test when the discovered topology information is matching with the deployed network topology or the discovered topology information for 3 consecutive queries return the same details.
6. Record the time last discovery message (T_{mn}) sent to controller from the forwarding plane test emulator interface (I1) when the test completed successfully. (e.g., the topology matches).

Measurement:

Topology Discovery Time $Tr1 = T_{mn} - T_{m1}$.

$$\text{Average Topology Discovery Time} = \frac{Tr1 + Tr2 + Tr3 \dots Trn}{\text{Total Test Iterations}}$$

Reporting Format:

The Topology Discovery Time results MUST be reported in the format of a table, with a row for each successful iteration. The last row of the table indicates the average Topology Discovery Time.

If this test is repeated with varying number of nodes over the same topology, the results SHOULD be reported in the form of a graph. The X coordinate SHOULD be the Number of nodes (N), the Y coordinate SHOULD be the average Topology Discovery Time.

If this test is repeated with same number of nodes over different topologies, the results SHOULD be reported in the form of a graph. The X coordinate SHOULD be the Topology Type, the Y coordinate SHOULD be the average Topology Discovery Time.

6.1.2 Asynchronous Message Processing Time

Objective:

Measure the time taken by the SDN controller to process an asynchronous message, expressed in milliseconds.

Reference Test Setup:

This test SHOULD use one of the test setup described in section 3.1 or section 3.2 of this document.

Prerequisite:

1. The controller MUST have completed the network topology discovery for the connected SDN nodes.

Procedure:

1. Generate asynchronous messages from every connected SDN node, to the SDN controller, one at a time in series from the forwarding plane test emulator for the test duration.
2. Record every request transmit (T1) timestamp and the corresponding response (R1) received timestamp at the forwarding plane test emulator interface (I1) for every successful message exchange.

Measurement:

$$\text{Asynchronous Message Processing Time } Tr1 = \frac{(R1-T1) + (R2-T2) \dots (Rn-Tn)}{Nrx}$$

Where Nrx is the total number of successful messages exchanged

$$\text{Average Asynchronous Message Processing Time} = \frac{Tr1 + Tr2 + Tr3 \dots Trn}{\text{Total Test Iterations}}$$

Reporting Format:

The Asynchronous Message Processing Time results MUST be reported in the format of a table with a row for each iteration. The last row of the table indicates the average Asynchronous Message Processing Time.

The report should capture the following information in addition to the configuration parameters captured in section 5.

- Successful messages exchanged (Nrx)

If this test is repeated with varying number of nodes with same topology, the results SHOULD be reported in the form of a graph. The X coordinate SHOULD be the Number of nodes (N), the Y coordinate SHOULD be the average Asynchronous Message Processing Time.

If this test is repeated with same number of nodes using different topologies, the results SHOULD be reported in the form of a graph. The X coordinate SHOULD be the Topology Type, the Y coordinate SHOULD be the average Asynchronous Message Processing Time.

6.1.3 Asynchronous Message Processing Rate

Objective:

To measure the maximum number of asynchronous messages (session aliveness check message, new flow arrival notification message etc.) a controller can process within the test duration, expressed in messages processed per second.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document.

Prerequisite:

1. The controller MUST have completed the network topology discovery for the connected SDN nodes.

Procedure:

1. Generate asynchronous messages continuously at the maximum possible rate on the established connections from all the connected SDN nodes in the forwarding plane test emulator for the Test Duration (Td).
2. Record total number of responses received from the controller (Nrx) as well as the number of messages sent (Ntx) to the controller within the test duration (Td) at the forwarding plane test emulator interface (I1) .

Measurement:

$$\text{Asynchronous Message Processing Rate } Tr1 = \frac{Nrx}{Td}$$

$$\text{Average Asynchronous Message Processing Rate} = \frac{Tr1 + Tr2 + Tr3 \dots Trn}{\text{Total Test Iterations}}$$

$$\text{Loss Ratio} = (Ntx - Nrx) / 100.$$

Reporting Format:

The Asynchronous Message Processing Rate results MUST be reported in the format of a table with a row for each iteration. The last row of the table indicates the average Asynchronous Message Processing Rate.

The report should capture the following information in addition to the configuration parameters captured in section 5.

- Offered rate (Ntx)
- Loss Ratio

If this test is repeated with varying number of nodes over same topology, the results SHOULD be reported in the form of a graph. The X coordinate SHOULD be the Number of nodes (N), the Y coordinate SHOULD be the average Asynchronous Message Processing Rate.

If this test is repeated with same number of nodes over different topologies, the results SHOULD be reported in the form of a graph. The X coordinate SHOULD be the Topology Type, the Y coordinate SHOULD be the average Asynchronous Message Processing Rate.

6.1.4 Reactive Path Provisioning Time

Objective:

To measure the time taken by the controller to setup a path reactively between source and destination node, expressed in milliseconds.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document.

Prerequisite:

1. The controller MUST contain the network topology information for the deployed network topology.
2. The controller should have the knowledge about the location of destination endpoint for which the path has to be provisioned. This can be achieved through dynamic learning or static provisioning.
3. Ensure that the default action for flow miss in SDN node is 'send to controller'.
4. Ensure that each SDN node in a path requires the controller to make the forwarding decision while paving the entire path.

Procedure:

1. Send a single traffic stream from test traffic generator TP1 to test traffic generator TP2.
2. Record the time of the first flow provisioning request message sent to the controller(Tsfl) from the SDN node at the forwarding plane test emulator interface (I1).
3. Wait for the arrival of first traffic frame at the Traffic Endpoint TP2 or the expiry of test duration (Td).
4. Record the time of the last flow provisioning response message received from the controller(Tdfl) to the SDN node at the forwarding plane test emulator interface (I1).

Measurement:

Reactive Path Provisioning Time $Tr1 = Tdf1 - Tsfl$.

$$\text{Average Reactive Path Provisioning Time} = \frac{Tr1 + Tr2 + Tr3 \dots Trn}{\text{Total Test Iterations}}$$

Reporting Format:

The Reactive Path Provisioning Time results MUST be reported in the format of a table with a row for each iteration. The last row of the table indicates the Average Reactive Path Provisioning Time

The report should capture the following information in addition to the configuration parameters captured in section 5.

- Number of SDN nodes in the path

6.1.5 Proactive Path Provisioning Time**Objective:**

To measure the time taken by the controller to setup a path proactively between source and destination node, expressed in milliseconds.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document.

Prerequisite:

1. The controller MUST contain the network topology information for the deployed network topology.
2. The controller should have the knowledge about the location of destination endpoint for which the path has to be provisioned. This can be achieved through dynamic learning or static provisioning.
3. Ensure that the default action for flow miss in SDN node is 'drop'.

Procedure:

1. Send single traffic stream from test traffic generator TP1 to TP2.
2. Install the flow entries to reach from test traffic generator TP1 to the test traffic generator TP2 through controller's northbound or management interface.
3. Wait for the arrival of first traffic frame at the test traffic generator TP2 or the expiry of test duration (Td).
4. Record the time when proactive flow is provisioned in the Controller (Tsfl) at the management plane test emulator interface I2.

5. Record the time of the last flow provisioning message received from the controller(Tdfl) at the forwarding plane test emulator interface I1.

Measurement:

Proactive Flow Provisioning Time $Tr1 = Tdfl - Tsfl$.

$$\text{Average Proactive Path Provisioning Time} = \frac{Tr1 + Tr2 + Tr3 \dots Trn}{\text{Total Test Iterations}}$$

Reporting Format:

The Proactive Path Provisioning Time results MUST be reported in the format of a table with a row for each iteration. The last row of the table indicates the Average Proactive Path Provisioning Time.

The report should capture the following information in addition to the configuration parameters captured in section 5.

- Number of SDN nodes in the path

6.1.6 Reactive Path Provisioning Rate

Objective:

Measure the maximum number of independent paths a controller can concurrently establish between source and destination nodes reactively within the test duration, expressed in paths per second.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document.

Prerequisite:

1. The controller MUST contain the network topology information for the deployed network topology.
2. The controller should have the knowledge about the location of destination addresses for which the paths have to be provisioned. This can be achieved through dynamic learning or static provisioning.
3. Ensure that the default action for flow miss in SDN node is 'send to controller'.
4. Ensure that each SDN node in a path requires the controller to make the forwarding decision while paving the entire path.

Procedure:

1. Send traffic with unique source and destination addresses from test traffic generator TP1.
2. Record total number of unique traffic frames (Ndf) received at the test traffic generator TP2 within the test duration (Td).

Measurement:

$$\text{Reactive Path Provisioning Rate Tr1} = \frac{\text{Ndf}}{\text{Td}}$$

$$\text{Average Reactive Path Provisioning Rate} = \frac{\text{Tr1} + \text{Tr2} + \text{Tr3} \dots \text{Trn}}{\text{Total Test Iterations}}$$

Reporting Format:

The Reactive Path Provisioning Rate results MUST be reported in the format of a table with a row for each iteration. The last row of the table indicates the Average Reactive Path Provisioning Rate.

The report should capture the following information in addition to the configuration parameters captured in section 5.

- Number of SDN nodes in the path
- Offered rate

6.1.7 Proactive Path Provisioning Rate

Objective:

Measure the maximum number of independent paths a controller can concurrently establish between source and destination nodes proactively within the test duration, expressed in paths per second.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document.

Prerequisite:

1. The controller MUST contain the network topology information for the deployed network topology.
2. The controller should have the knowledge about the location of destination addresses for which the paths have to be provisioned. This can be achieved through dynamic learning or static provisioning.
3. Ensure that the default action for flow miss in SDN node is 'drop'.

Procedure:

1. Send traffic continuously with unique source and destination addresses from test traffic generator TP1.
2. Install corresponding flow entries to reach from simulated sources at the test traffic generator TP1 to the simulated destinations at test traffic generator TP2 through controller's northbound or management interface.
3. Record total number of unique traffic frames received Ndf) at the test traffic generator TP2 within the test duration (Td).

Measurement:

$$\text{Proactive Path Provisioning Rate Tr1} = \frac{\text{Ndf}}{\text{Td}}$$

$$\text{Average Proactive Path Provisioning Rate} = \frac{\text{Tr1} + \text{Tr2} + \text{Tr3} \dots \text{Trn}}{\text{Total Test Iterations}}$$

Reporting Format:

The Proactive Path Provisioning Rate results MUST be reported in the format of a table with a row for each iteration. The last row of the table indicates the Average Proactive Path Provisioning Rate.

The report should capture the following information in addition to the configuration parameters captured in section 5.

- Number of SDN nodes in the path
- Offered rate

6.1.8 Network Topology Change Detection Time**Objective:**

Measure the time taken by the controller to detect any changes in the network topology, expressed in milliseconds.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document.

Prerequisite:

1. The controller MUST have discovered the network topology information for the deployed network topology.
2. The periodic network discovery operation should be configured to twice the Test duration (Td) value.

Procedure:

1. Trigger a topology change event by bringing down an active SDN node in the topology.
2. Record the time when the first topology change notification is sent to the controller (Tcn) at the forwarding plane test emulator interface (I1).
3. Stop the test when the controller sends the first topology re-discovery message to the SDN node or the expiry of test interval (Td).
4. Record the time when the first topology re-discovery message is received from the controller (Tcd) at the forwarding plane test emulator interface (I1)

Measurement:

Network Topology Change Detection Time $Tr1 = Tcd - Tcn$.

$$\text{Average Network Topology Change Detection Time} = \frac{Tr1 + Tr2 + Tr3 \dots Trn}{\text{Total Test Iterations}}$$

Reporting Format:

The Network Topology Change Detection Time results MUST be reported in the format of a table with a row for each iteration. The last row of the table indicates the average Network Topology Change Time.

6.2 Scalability**6.2.1 Control Session Capacity****Objective:**

Measure the maximum number of control sessions that the controller can maintain.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document.

Procedure:

1. Establish control connection with controller from every SDN nodes emulated in the forwarding plane test emulator.
2. Stop the test when the controller starts dropping the control connection.
3. Record the number of successful connections established with the controller (CCn) at the forwarding plane test emulator.

Measurement:

Control Sessions Capacity = CCn.

Reporting Format:

The Control Session Capacity results MUST be reported in addition to the configuration parameters captured in section 5.

6.2.2 Network Discovery Size**Objective:**

Measure the network size (number of nodes, links, and hosts) that a controller can discover.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document.

Prerequisite:

1. The controller MUST support automatic network discovery.
2. Tester should be able to retrieve the discovered topology information either through controller's management interface or northbound interface.

Procedure:

1. Establish the network connections between controller and network nodes.
2. Query the controller for the discovered network topology information and compare it with the deployed network topology information.
- 3a. Increase the number of nodes by 1 when the comparison is successful and repeat the test.
- 3b. Decrease the number of nodes by 1 when the comparison fails and repeat the test.
4. Continue the test until the comparison of step 3b is successful.
5. Record the number of nodes for the last iteration (Ns) where the topology comparison was successful.

Measurement:

Network Discovery Size = Ns.

Reporting Format:

The Network Discovery Size results MUST be reported in addition to the configuration parameters captured in section 5.

6.2.3 Forwarding Table Capacity

Objective:

Measure the maximum number of flow entries a controller can manage in its Forwarding table.

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document.

Prerequisite:

1. The controller Forwarding table should be empty.
2. Flow Idle time MUST be set to higher or infinite value.
3. The controller MUST have completed network topology discovery.
4. Tester should be able to retrieve the forwarding table information either through controller's management interface or northbound interface.

Procedure:**Reactive Flow Provisioning Mode:**

1. Send bi-directional traffic continuously with unique source and/or destination addresses from test traffic generators TP1 and TP2 at the asynchronous message processing rate of controller.
2. Query the controller at a regular interval (e.g., 5 seconds) for the number of learnt flow entries from its northbound interface.
3. Stop the test when the retrieved value is constant for three consecutive iterations and record the value received from the last query (Nrp).

Proactive Flow Provisioning Mode:

1. Install unique flows continuously through controller's northbound or management interface until a failure response is received from the controller.
2. Record the total number of successful responses (Nrp).

Note:

Some controller designs for proactive flow provisioning mode may require the switch to send flow setup requests in order to generate flow setup responses. In such cases, it is recommended to generate bi-directional traffic for the provisioned flows.

Measurement:

Proactive Flow Provisioning Mode:

Max Flow Entries = Total number of flows provisioned (Nrp)

Reactive Flow Provisioning Mode:

Max Flow Entries = Total number of learnt flow entries (Nrp)

Forwarding Table Capacity = Max Flow Entries.

Reporting Format:

The Forwarding Table Capacity results MUST be tabulated with the following information in addition to the configuration parameters captured in section 5.

- Provisioning Type (Proactive/Reactive)

6.3 Security

6.3.1 Exception Handling

Objective:

Determine the effect of handling error packets and notifications on performance tests. The impact MUST be measured for the following performance tests

- a. Path Provisioning Rate
- b. Path Provisioning Time
- c. Network Topology Change Detection Time

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document.

Prerequisite:

1. This test MUST be performed after obtaining the baseline measurement results for the above performance tests.
2. Ensure that the invalid messages are not dropped by the intermediate devices connecting the controller and SDN nodes.

Procedure:

1. Perform the above listed performance tests and send 1% of messages from the Asynchronous Message Processing Rate as invalid messages from the connected SDN nodes emulated at the forwarding plane test emulator.

2. Perform the above listed performance tests and send 2% of messages from the Asynchronous Message Processing Rate as invalid messages from the connected SDN nodes emulated at the forwarding plane test emulator.

Note:

1. Invalid messages can be frames with incorrect protocol fields or any form of failure notifications sent towards controller.

Measurement:

Measurement MUST be done as per the equation defined in the corresponding performance test measurement section.

Reporting Format:

The Exception Handling results MUST be reported in the format of table with a column for each of the below parameters and row for each of the listed performance tests.

- Without Exceptions
- With 1% Exceptions
- With 2% Exceptions

6.3.2 Denial of Service Handling

Objective:

Determine the effect of handling DoS attacks on performance and scalability tests the impact MUST be measured for the following tests:

- a. Path Provisioning Rate
- b. Path Provisioning Time
- c. Network Topology Change Detection Time
- d. Network Discovery Size

Reference Test Setup:

The test SHOULD use one of the test setups described in section 3.1 or section 3.2 of this document.

Prerequisite:

This test MUST be performed after obtaining the baseline measurement results for the above tests.

Procedure:

1. Perform the listed tests and launch a DoS attack towards controller while the test is running.

Note:

DoS attacks can be launched on one of the following interfaces.

- a. Northbound (e.g., Sending a huge number of requests on northbound interface)
- b. Management (e.g., Ping requests to controller's management interface)
- c. Southbound (e.g., TCP SYNC messages on southbound interface)

Measurement:

Measurement MUST be done as per the equation defined in the corresponding test's measurement section.

Reporting Format:

The DoS Attacks Handling results MUST be reported in the format of table with a column for each of the below parameters and row for each of the listed tests.

- Without any attacks
- With attacks

The report should also specify the nature of attack and the interface.

6.4 Reliability

6.4.1 Controller Failover Time

Objective:

Measure the time taken to switch from an active controller to the backup controller, when the controllers work in redundancy mode and the active controller fails.

Reference Test Setup:

The test SHOULD use the test setup described in section 3.2 of this document.

Prerequisite:

1. Master controller election MUST be completed.
2. Nodes are connected to the controller cluster as per the Redundancy Mode (RM).
3. The controller cluster should have completed the network topology discovery.
4. The SDN Node MUST send all new flows to the controller when it receives from the test traffic generator.
5. Controller should have learnt the location of destination (D1) at test traffic generator TP2.

Procedure:

1. Send uni-directional traffic continuously with incremental sequence number and source addresses from test traffic generator TP1 at the rate that the controller processes without any drops.
2. Ensure that there are no packet drops observed at the test traffic generator TP2.
3. Bring down the active controller.
4. Stop the test when a first frame received on TP2 after failover operation.
5. Record the time at which the last valid frame received (T1) at test traffic generator TP2 before sequence error and the first valid frame received (T2) after the sequence error at TP2

Measurement:

Controller Failover Time = (T2 - T1)

Packet Loss = Number of missing packet sequences.

Reporting Format:

The Controller Failover Time results MUST be tabulated with the following information.

- Number of cluster nodes
- Redundancy mode
- Controller Failover
- Time Packet Loss
- Cluster keep-alive interval

6.4.2 Network Re-Provisioning Time

Objective:

Compute the time taken to re-route the traffic by the controller when there is a failure in existing traffic paths.

Reference Test Setup:

This test SHOULD use one of the test setup described in section 3.1 or section 3.2 of this document.

Prerequisite:

1. Network with the given number of nodes and redundant paths MUST be deployed.
2. Ensure that the controller MUST have knowledge about the location of test traffic generators TP1 and TP2.
3. Ensure that the controller does not pre-provision the alternate path in the emulated SDN nodes at the forwarding plane test emulator.

Procedure:

1. Send bi-directional traffic continuously with unique sequence number from TP1 and TP2.
2. Bring down a link or switch in the traffic path.
3. Stop the test after receiving first frame after network re-convergence.
4. Record the time of last received frame prior to the frame loss at TP2 (TP2-Tlfr) and the time of first frame received after the frame loss at TP2 (TP2-Tffr).
5. Record the time of last received frame prior to the frame loss at TP1 (TP1-Tlfr) and the time of first frame received after the frame loss at TP1 (TP1-Tffr).

Measurement:

Forward Direction Path Re-Provisioning Time (FDRT)
= (TP2-Tffr - TP2-Tlfr)

Reverse Direction Path Re-Provisioning Time (RDRT)
= (TP1-Tffr - TP1-Tlfr)

Network Re-Provisioning Time = (FDRT+RDRT)/2

Forward Direction Packet Loss = Number of missing sequence frames at TP1

Reverse Direction Packet Loss = Number of missing sequence frames at TP2

Reporting Format:

The Network Re-Provisioning Time results MUST be tabulated with the following information.

- Number of nodes in the primary path
- Number of nodes in the alternate path
- Network Re-Provisioning Time
- Forward Direction Packet Loss
- Reverse Direction Packet Loss

7. References

7.1 Normative References

- [RFC2544] S. Bradner, J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, March 1999.
- [RFC2330] V. Paxson, G. Almes, J. Mahdavi, M. Mathis, "Framework for IP Performance Metrics", RFC 2330, May 1998.

- [RFC6241] R. Enns, M. Bjorklund, J. Schoenwaelder, A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, July 2011.
- [RFC6020] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010
- [RFC5440] JP. Vasseur, JL. Le Roux, "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, March 2009.
- [OpenFlow Switch Specification] ONF, "OpenFlow Switch Specification" Version 1.4.0 (Wire Protocol 0x05), October 14, 2013.
- [I-D.sdn-controller-benchmark-term] Bhuvaneswaran.V, Anton Basil, Mark.T, Vishwas Manral, Sarah Banks "Terminology for Benchmarking SDN Controller Performance", draft-bhuvan-bmwg-sdn-controller-benchmark-term-00 (Work in progress), March 23, 2015
- [I-D.i2rs-architecture] A. Atlas, J. Halpern, S. Hares, D. Ward, T. Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture-09 (Work in progress), March 6, 2015

7.2 Informative References

- [OpenContrail] Ankur Singla, Bruno Rijsman, "OpenContrail Architecture Documentation",
<http://opencontrail.org/opencontrail-architecture-documentation>
- [OpenDaylight] OpenDaylight Controller:Architectural Framework,
https://wiki.opendaylight.org/view/OpenDaylight_Controller

8. IANA Considerations

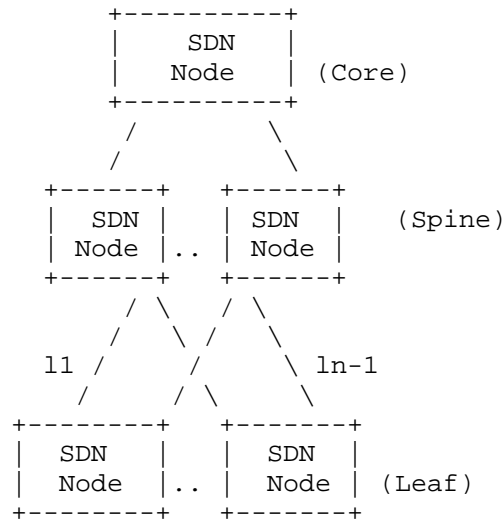
This document does not have any IANA requests.

9. Security Considerations

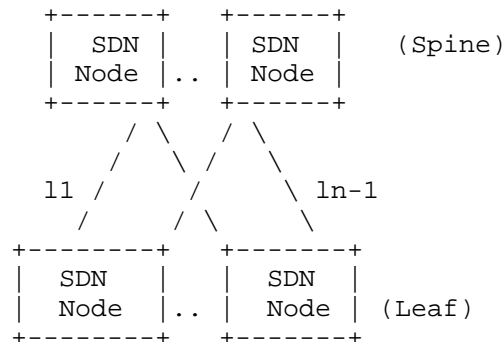
Benchmarking tests described in this document are limited to the performance characterization of controller in lab environment with isolated network.

10. Appendix A - Example Test Topologies

10.1. Leaf-Spine Topology - Three Tier Network Architecture



10.2. Leaf-Spine Topology - Two Tier Network Architecture



11. Appendix A - Benchmarking Methodology using OpenFlow(OF) Controllers

This section gives an overview of OpenFlow protocol and provides test methodology to benchmark SDN controllers supporting OpenFlow southbound protocol.

11.1. Protocol Overview

OpenFlow is an open standard protocol defined by Open Networking Foundation (ONF), used for programming the forwarding plane of network switches or routers via a centralized controller.

11.2. Messages Overview

OpenFlow protocol supports three messages types namely controller-to-switch, asynchronous and symmetric.

Controller-to-switch messages are initiated by the controller and used to directly manage or inspect the state of the switch. These messages allow controllers to query/configure the switch (Features, Configuration messages), collect information from switch (Read-State message), send packets on specified port of switch (Packet-out message), and modify switch forwarding plane and state (Modify-State, Role-Request messages etc.).

Asynchronous messages are generated by the switch without a controller soliciting them. These messages allow switches to update controllers to denote an arrival of new flow (Packet-in), switch state change (Flow-Removed, Port-status) and error (Error).

Symmetric messages are generated in either direction without solicitation. These messages allow switches and controllers to set up connection (Hello), verify for liveness (Echo) and offer additional functionalities (Experimenter).

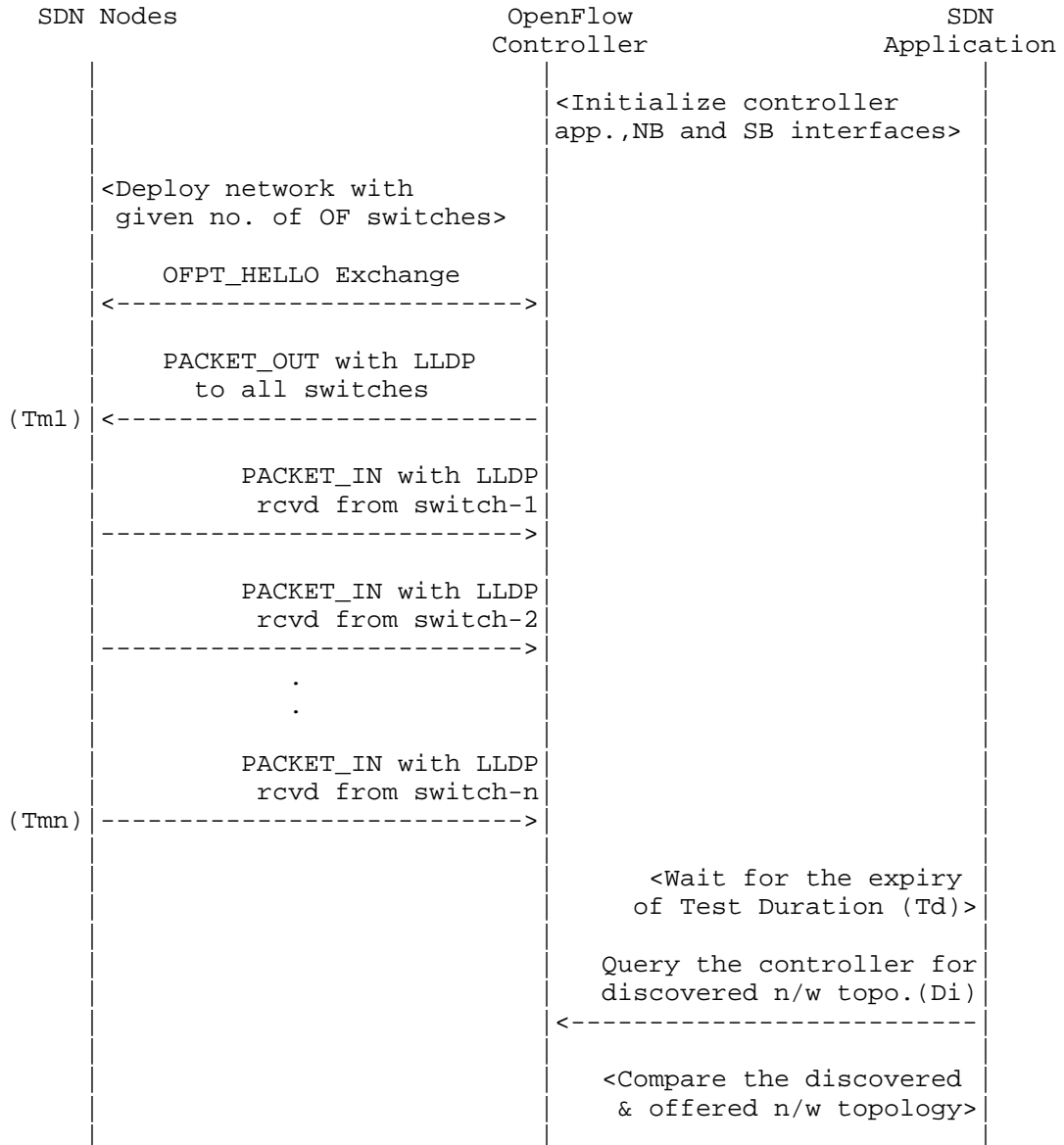
11.3. Connection Overview

OpenFlow channel is used to exchange OpenFlow message between an OpenFlow switch and an OpenFlow controller. The OpenFlow channel connection can be setup using plain TCP or TLS. By default, a switch establishes single connection with SDN controller. A switch may establish multiple parallel connections to single controller (auxiliary connection) or multiple controllers to handle controller failures and load balancing.

11.4 Performance Benchmarking Tests

11.4.1 Network Topology Discovery Time

Procedure:



Legend:

NB: Northbound

SB: Southbound

OF: OpenFlow

Tm1: Time of reception of first LLDP message from controller

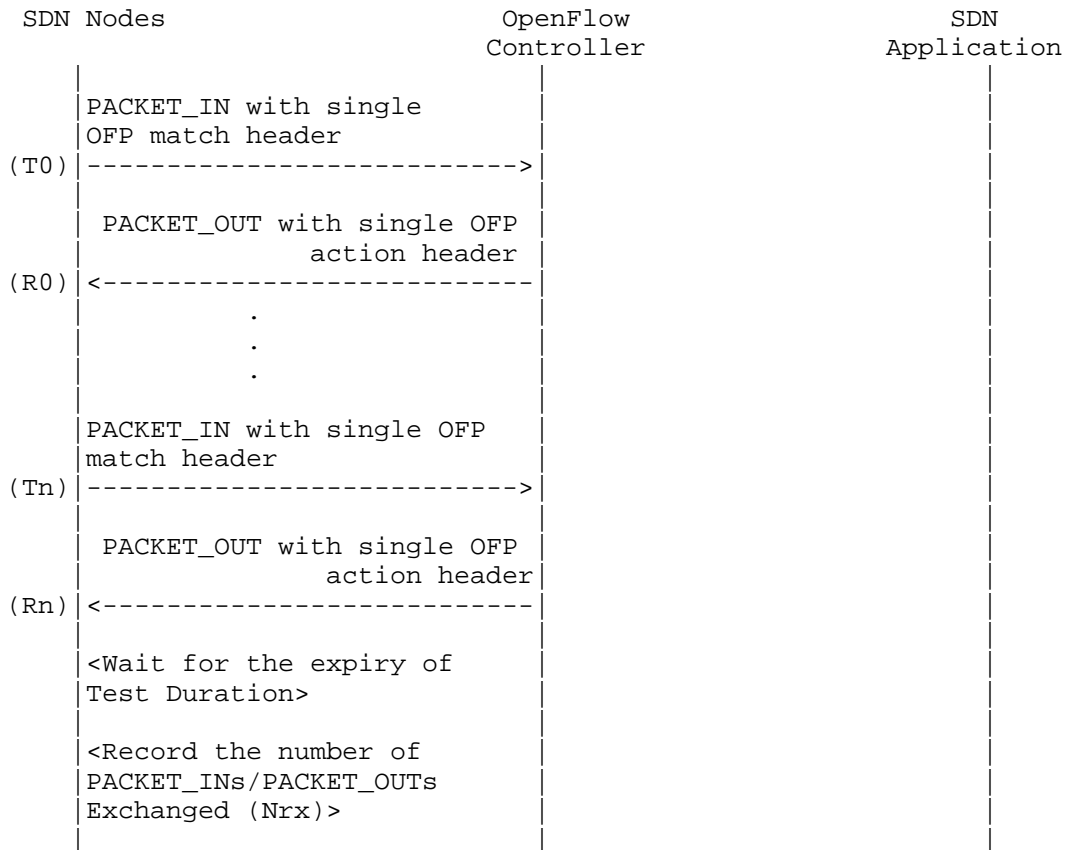
Tmn: Time of last LLDP message sent to controller

Discussion:

The Network Topology Discovery Time can be obtained by calculating the time difference between the first PACKET_OUT with LLDP message received from the controller (Tm1) and the last PACKET_IN with LLDP message sent to the controller (Tmn) when the comparison is successful.

11.4.2 Asynchronous Message Processing Time

Procedure:



Legend:

T0,T1, ..Tn are PACKET_IN messages transmit timestamps.

R0,R1, ..Rn are PACKET_OUT messages receive timestamps.

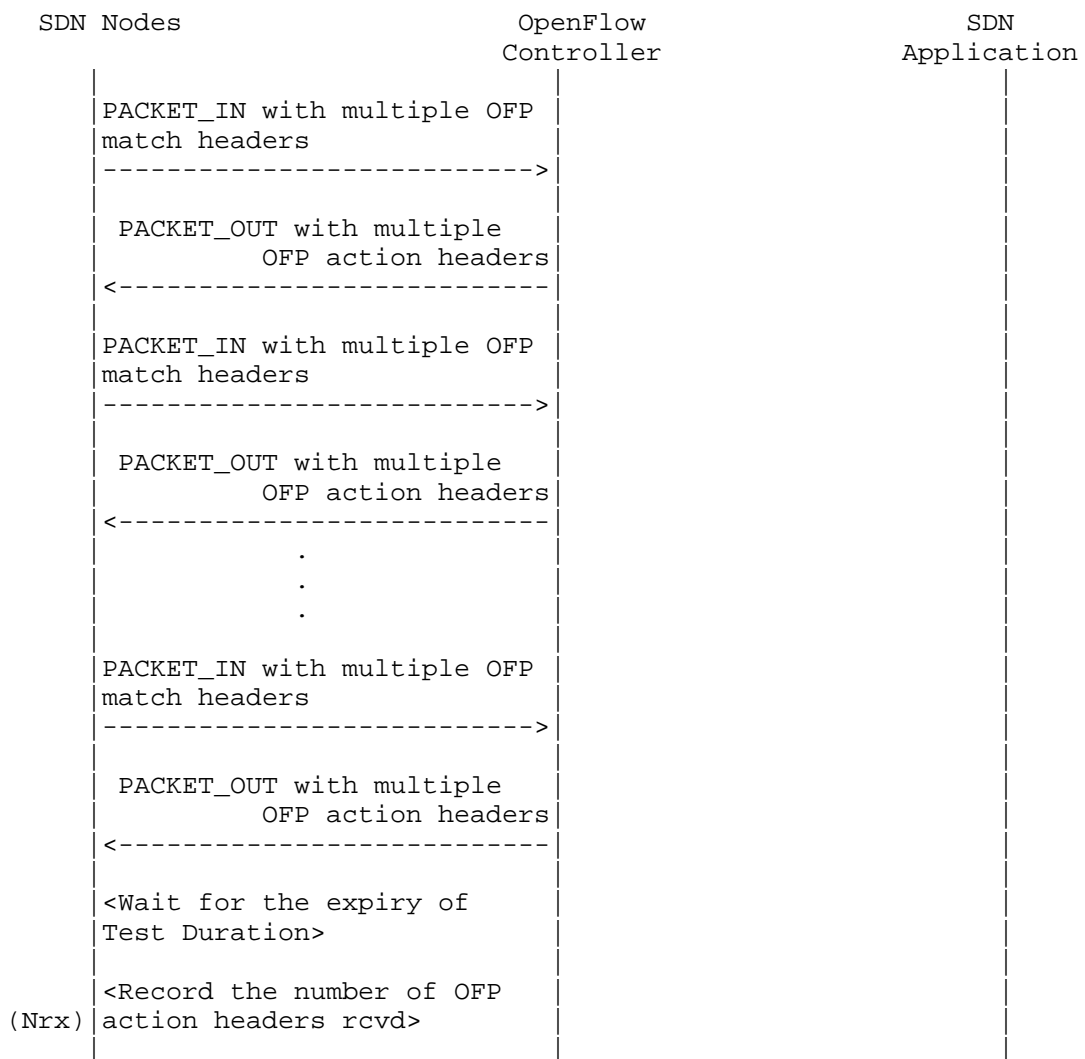
Nrx : Number of successful PACKET_IN/PACKET_OUT message exchanges

Discussion:

The Asynchronous Message Processing Time will be obtained by sum of $((R0-T0),(R1-T1)..(Rn - Tn))/ Nrx$.

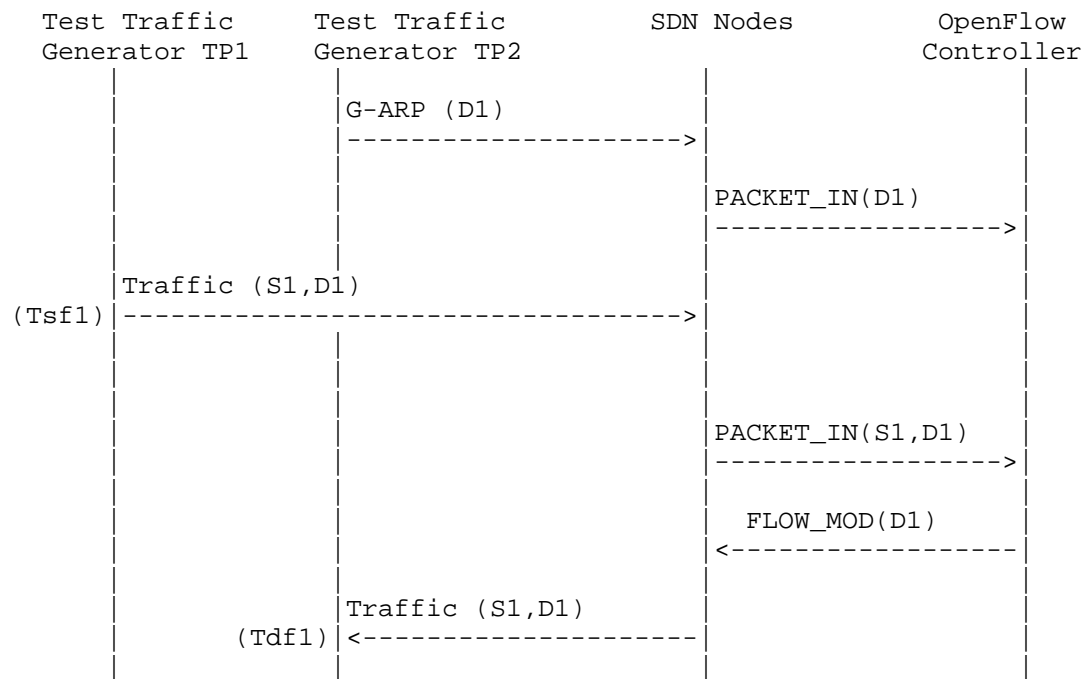
11.4.3 Asynchronous Message Processing Rate

Procedure:



Discussion:

The Asynchronous Message Processing Rate will be obtained by calculating the number of OFP action headers received in all PACKET_OUT messages during the test duration.

11.4.4 Reactive Path Provisioning Time**Procedure:****Legend:**

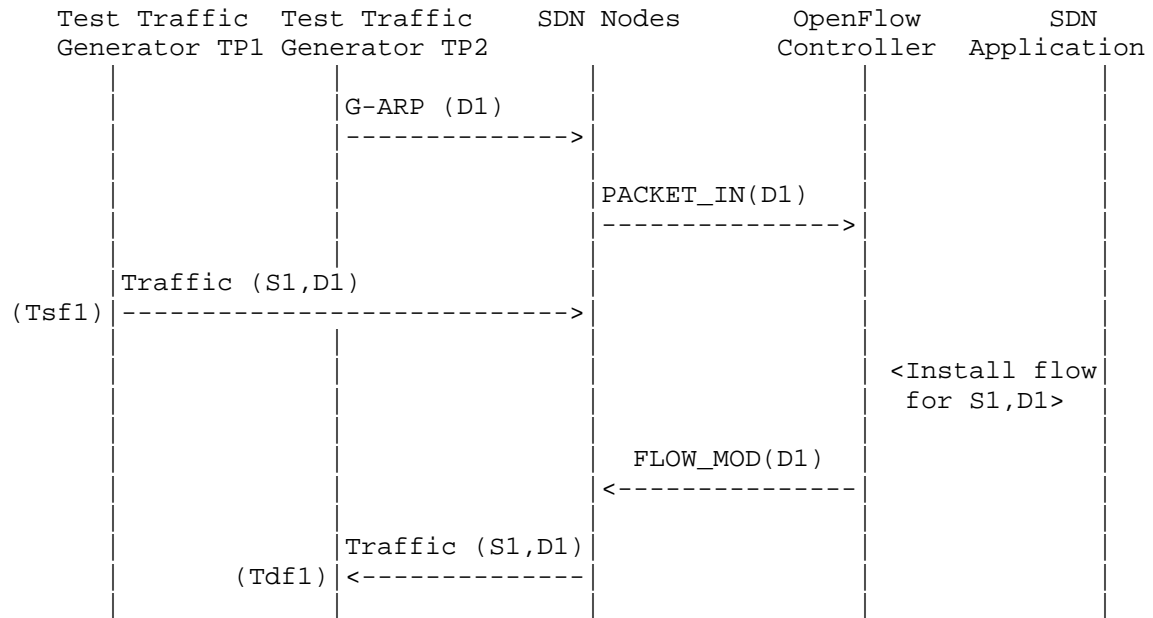
G-ARP: Gratuitous ARP message.
 Tsfl: Time of first frame sent from TP1
 Tdf1: Time of first frame received from TP2

Discussion:

The Reactive Path Provisioning Time can be obtained by finding the time difference between the transmit and receive time of the traffic (Tsfl-Tdf1).

11.4.5 Proactive Path Provisioning Time

Procedure:



Legend:

G-ARP: Gratuitous ARP message.

Tsfl: Time of first frame sent from TP1

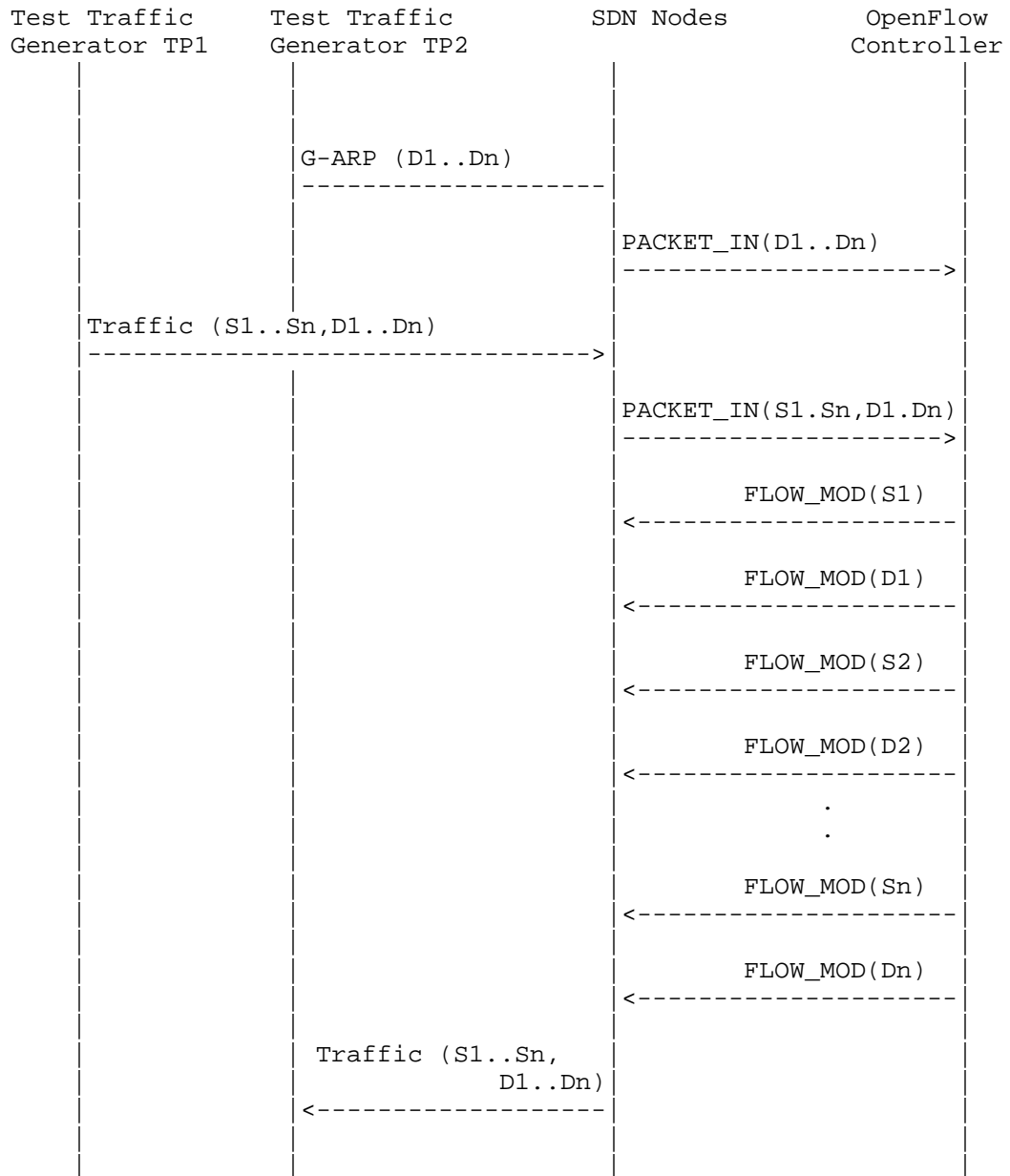
Tdfl: Time of first frame received from TP2

Discussion:

The Proactive Path Provisioning Time can be obtained by finding the time difference between the transmit and receive time of the traffic (Tsfl-Tdfl).

11.4.6 Reactive Path Provisioning Rate

Procedure:



Legend:

G-ARP: Gratuitous ARP

D1..Dn: Destination Endpoint 1, Destination Endpoint 2
Destination Endpoint n

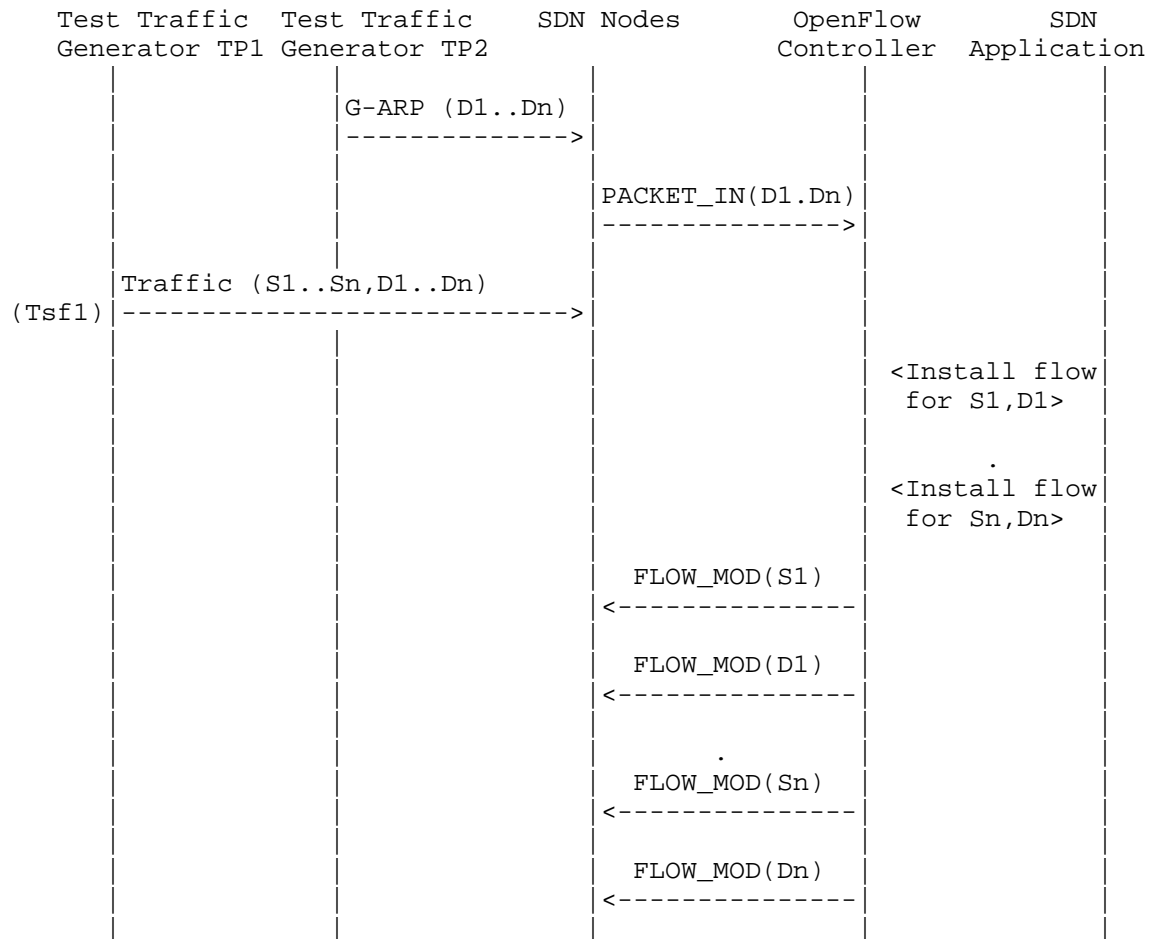
S1..Sn: Source Endpoint 1, Source Endpoint 2 .., Source Endpoint n

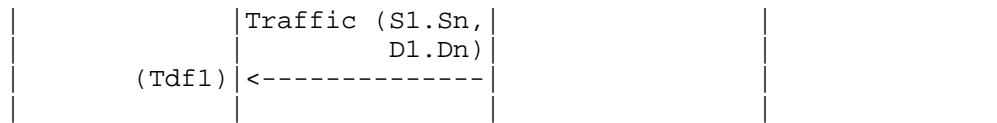
Discussion:

The Reactive Path Provisioning Rate can be obtained by finding the total number of frames received at TP2 after the test duration.

11.4.7 Proactive Path Provisioning Rate

Procedure:





Legend:

G-ARP: Gratuitous ARP

D1..Dn: Destination Endpoint 1, Destination Endpoint 2
Destination Endpoint n

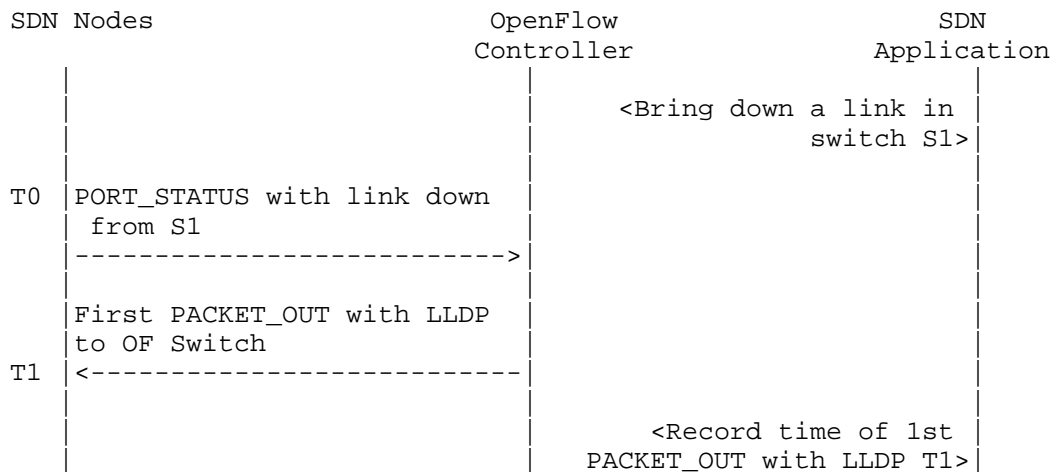
S1..Sn: Source Endpoint 1, Source Endpoint 2 .., Source Endpoint n

Discussion:

The Proactive Path Provisioning Rate can be obtained by finding the total number of frames received at TP2 after the test duration

11.4.8 Network Topology Change Detection Time

Procedure:



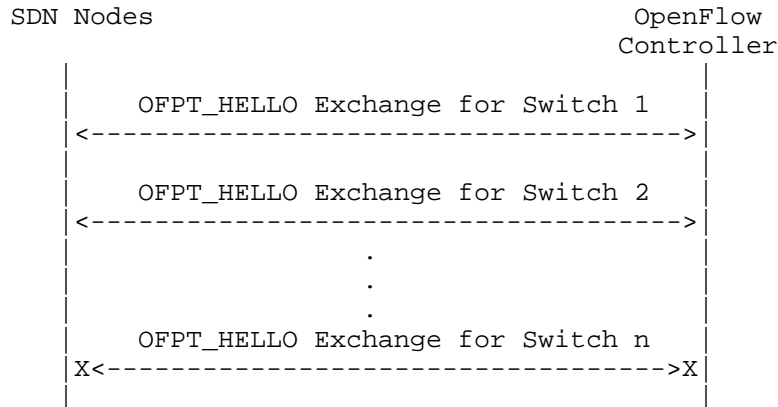
Discussion:

The Network Topology Change Detection Time can be obtained by finding the difference between the time the OpenFlow switch S1 sends the PORT_STATUS message (T0) and the time that the OpenFlow controller sends the first topology re-discovery message (T1) to OpenFlow switches.

11.5 Scalability

11.5.1 Control Sessions Capacity

Procedure:

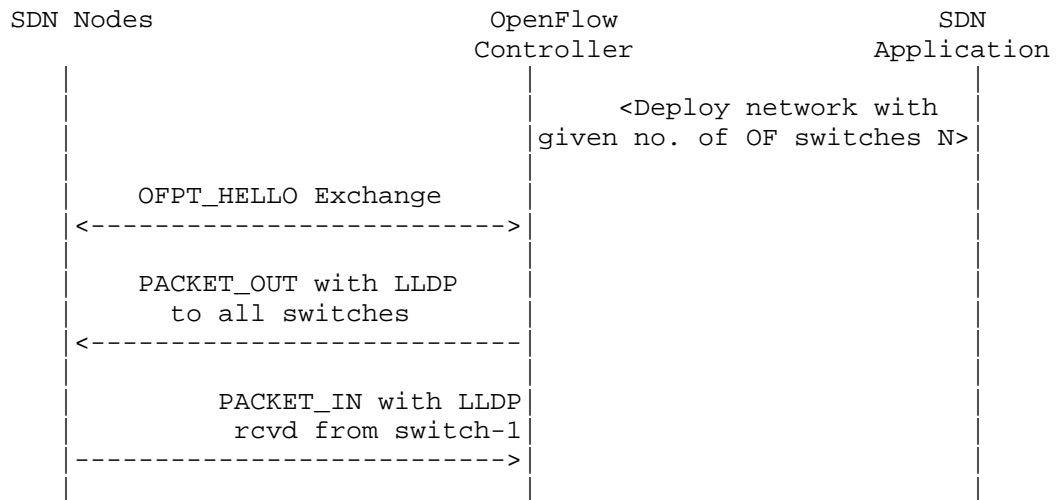


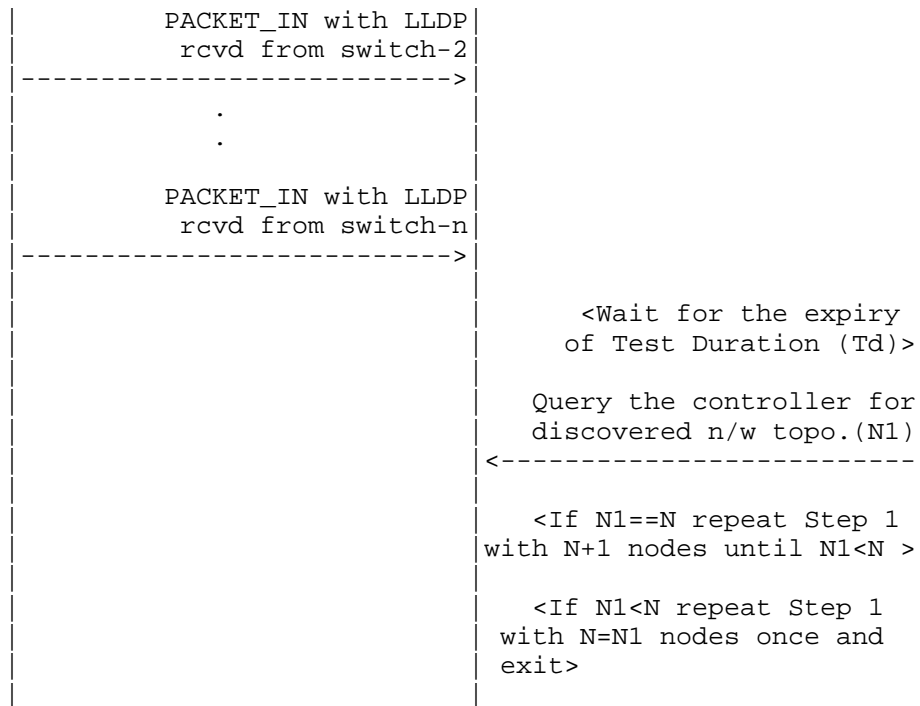
Discussion:

The value of Switch n-1 will provide Control Sessions Capacity.

11.5.2 Network Discovery Size

Procedure:

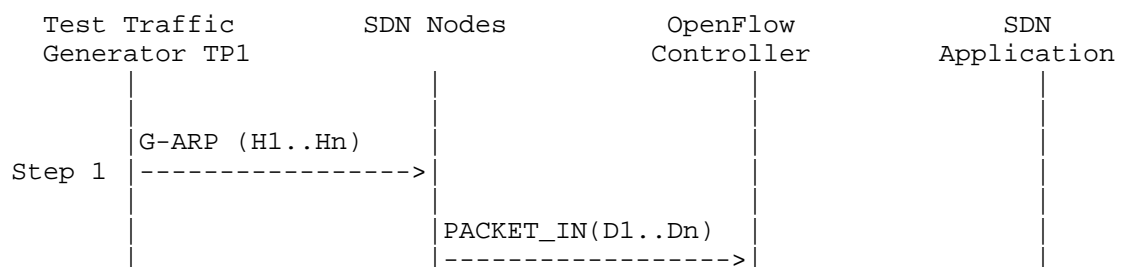


**Legend:**

n/w topo: Network Topology
OF: OpenFlow

Discussion:

The value of N1 provides the Network Discovery Size value. The test duration can be set to the stipulated time within which the user expects the controller to complete the discovery process.

11.5.3 Forwarding Table Capacity**Procedure:**

Step 2			<Wait for 5 secs>	
			<Query for FWD entry>	(F1)
			<Wait for 5 secs>	
			<Query for FWD entry>	(F2)
			<Wait for 5 secs>	
			<Query for FWD entry>	(F3)
			<Repeat Step 2 until F1==F2==F3>	

Legend:

G-ARP: Gratuitous ARP
H1..Hn: Host 1 .. Host n
FWD: Forwarding Table

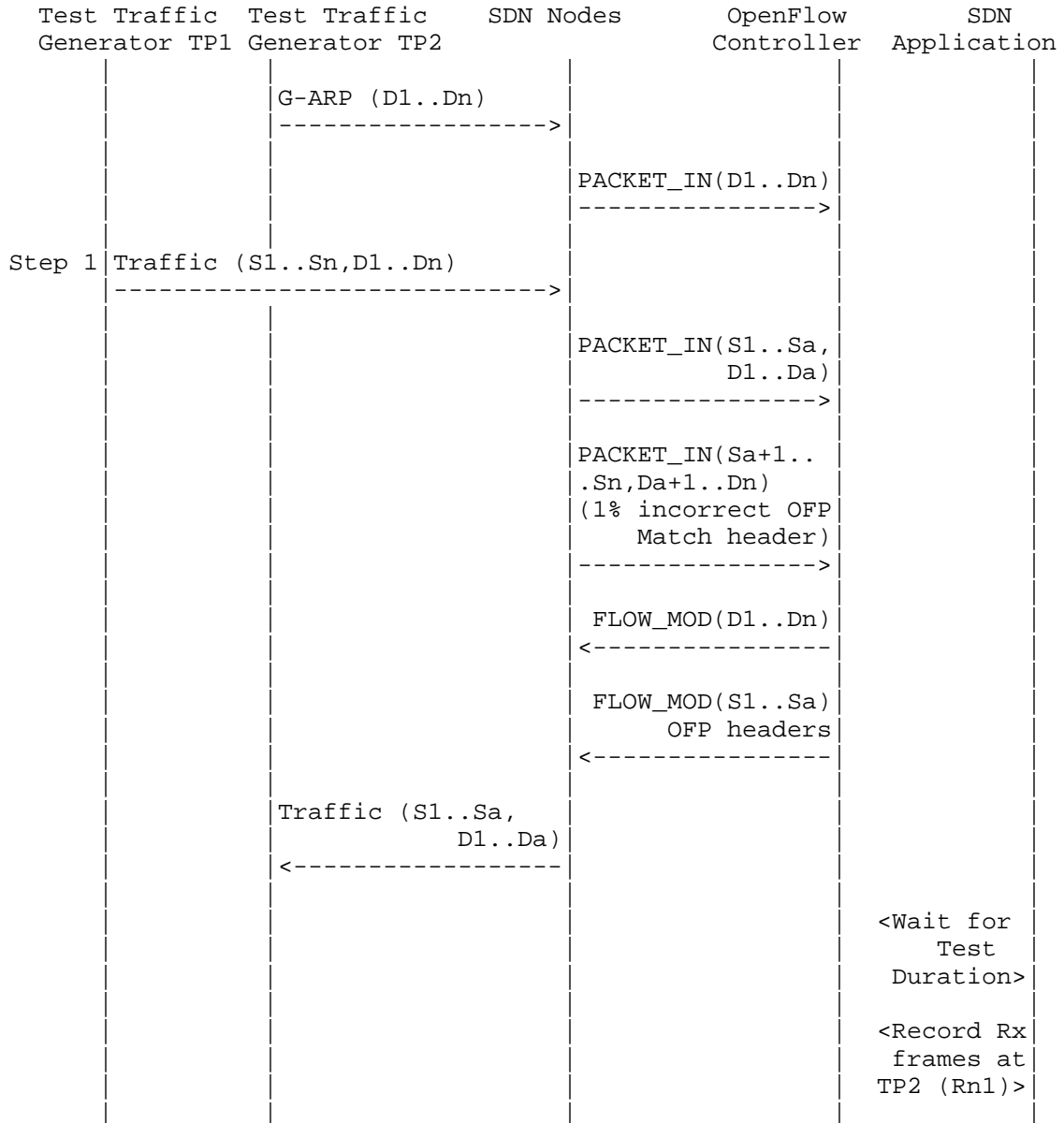
Discussion:

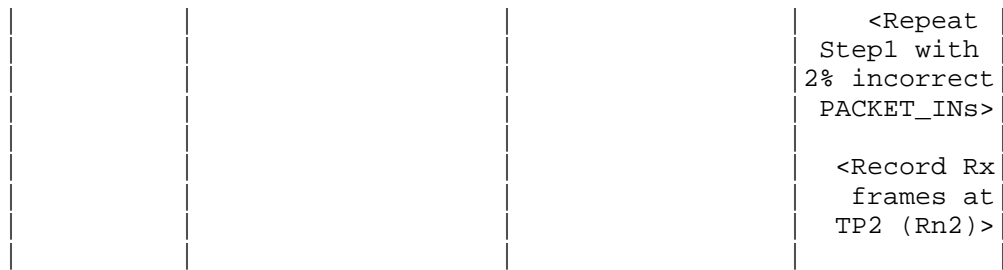
Query the controller forwarding table entries for multiple times until the three consecutive queries return the same value. The last value retrieved from the controller will provide the Forwarding Table Capacity value. The query interval is user configurable. The 5 seconds shown in this example is for representational purpose.

11.6 Security

11.6.1 Exception Handling

Procedure:



**Legend:**

G-ARP: Gratuitous ARP

PACKET_IN(Sa+1..Sn,Da+1..Dn): OpenFlow PACKET_IN with wrong version number

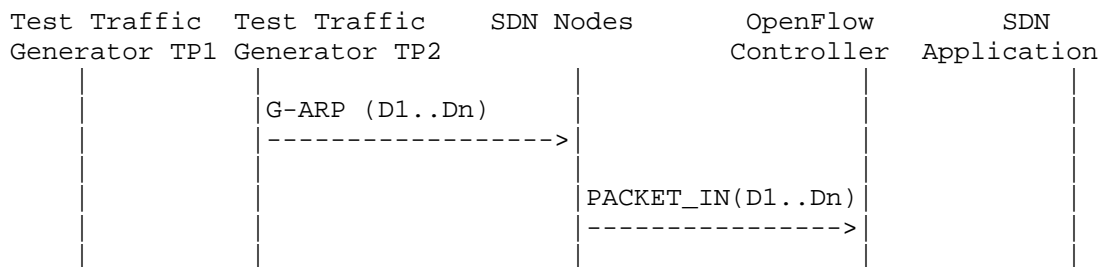
Rn1: Total number of frames received at Test Port 2 with 1% incorrect frames

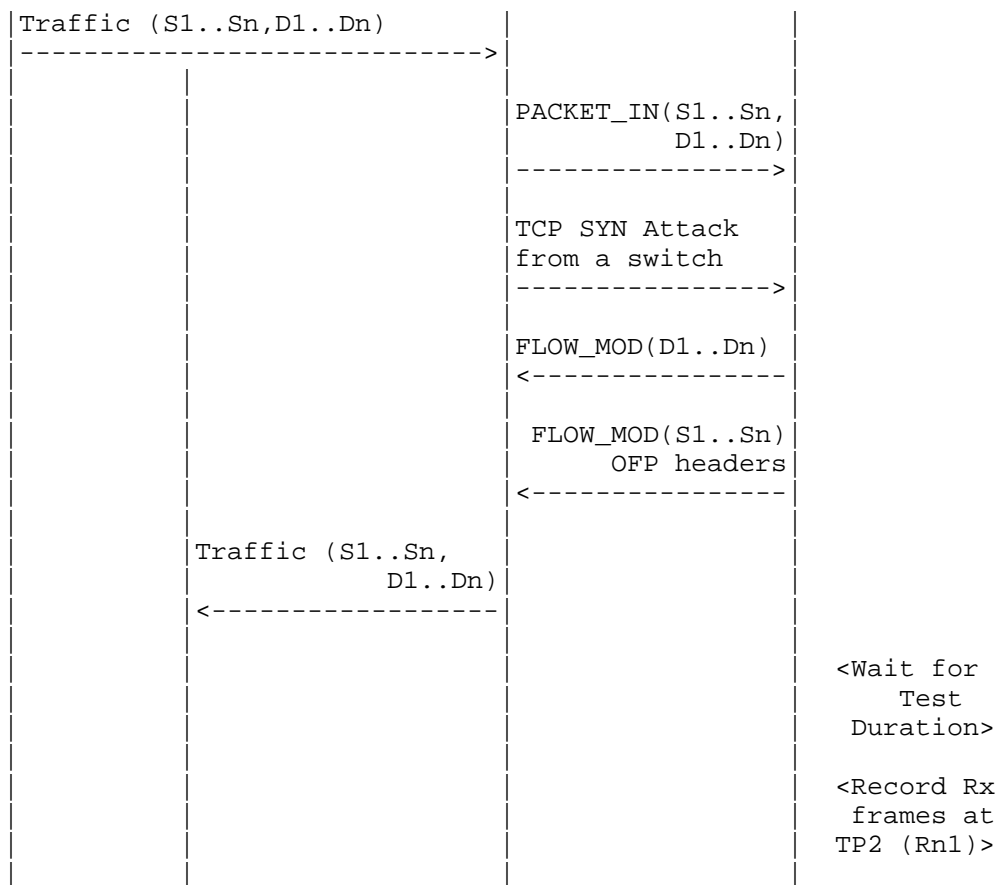
Rn2: Total number of frames received at Test Port 2 with 2% incorrect frames

Discussion:

The traffic rate sent towards OpenFlow switch from Test Port 1 should be 1% higher than the Path Programming Rate. Rn1 will provide the Path Provisioning Rate of controller at 1% of incorrect frames handling and Rn2 will provide the Path Provisioning Rate of controller at 2% of incorrect frames handling.

The procedure defined above provides test steps to determine the effect of handling error packets on Path Programming Rate. Same procedure can be adopted to determine the effects on other performance tests listed in this benchmarking tests.

11.6.2 Denial of Service Handling**Procedure:**



Legend:

G-ARP: Gratuitous ARP

Discussion:

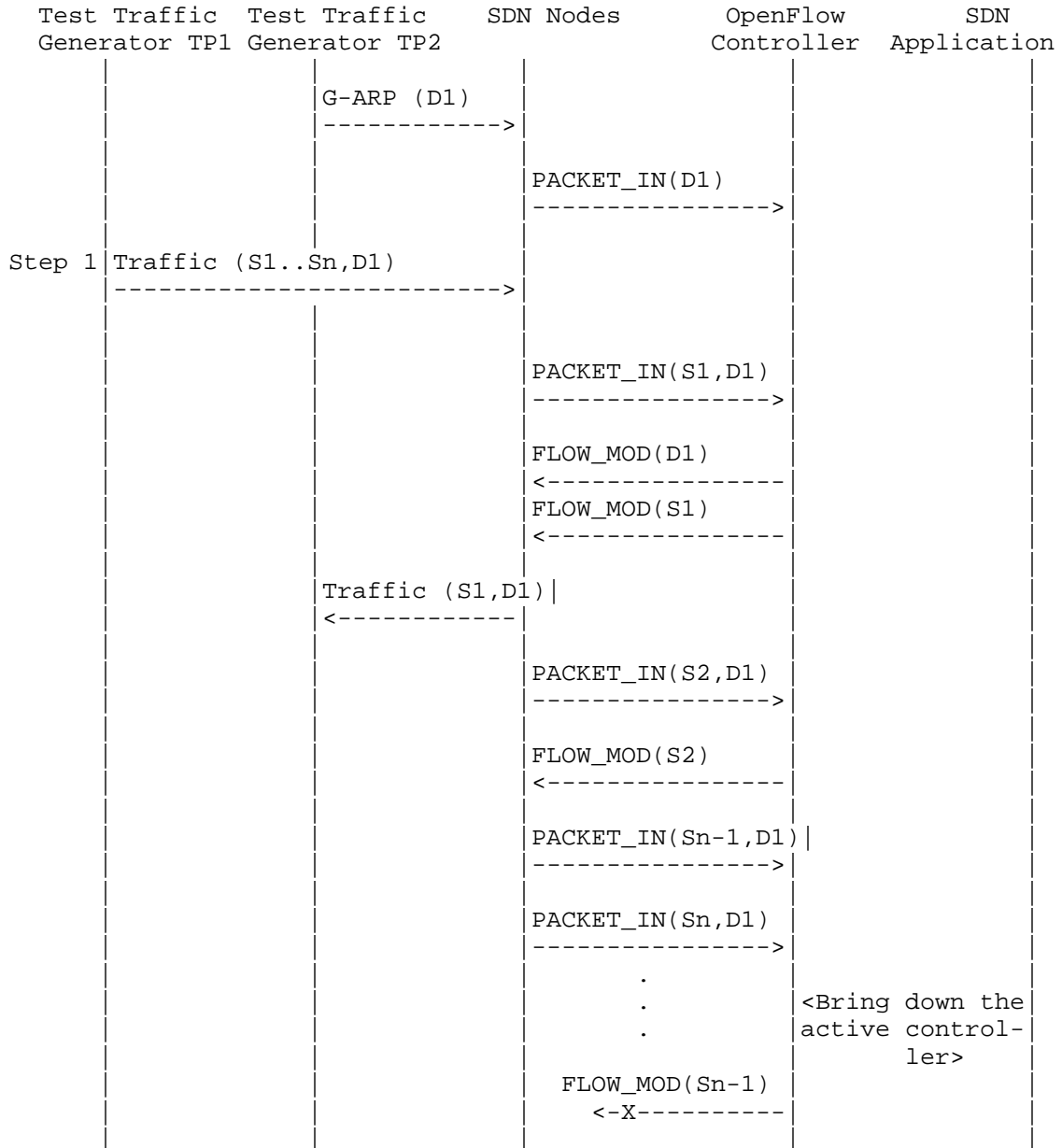
TCP SYN attack should be launched from one of the emulated/simulated OpenFlow Switch. Rn1 provides the Path Programming Rate of controller upon handling denial of service attack.

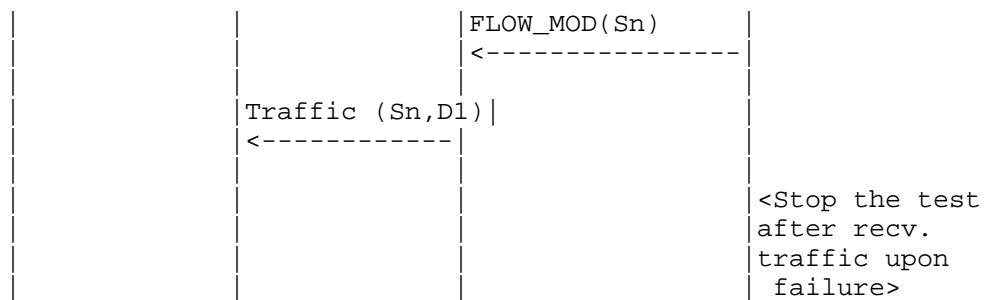
The procedure defined above provides test steps to determine the effect of handling denial of service on Path Programming Rate. Same procedure can be adopted to determine the effects on other performance tests listed in this benchmarking tests.

11.7 Reliability

11.7.1 Controller Failover Time

Procedure:





Legend:

G-ARP: Gratuitous ARP.

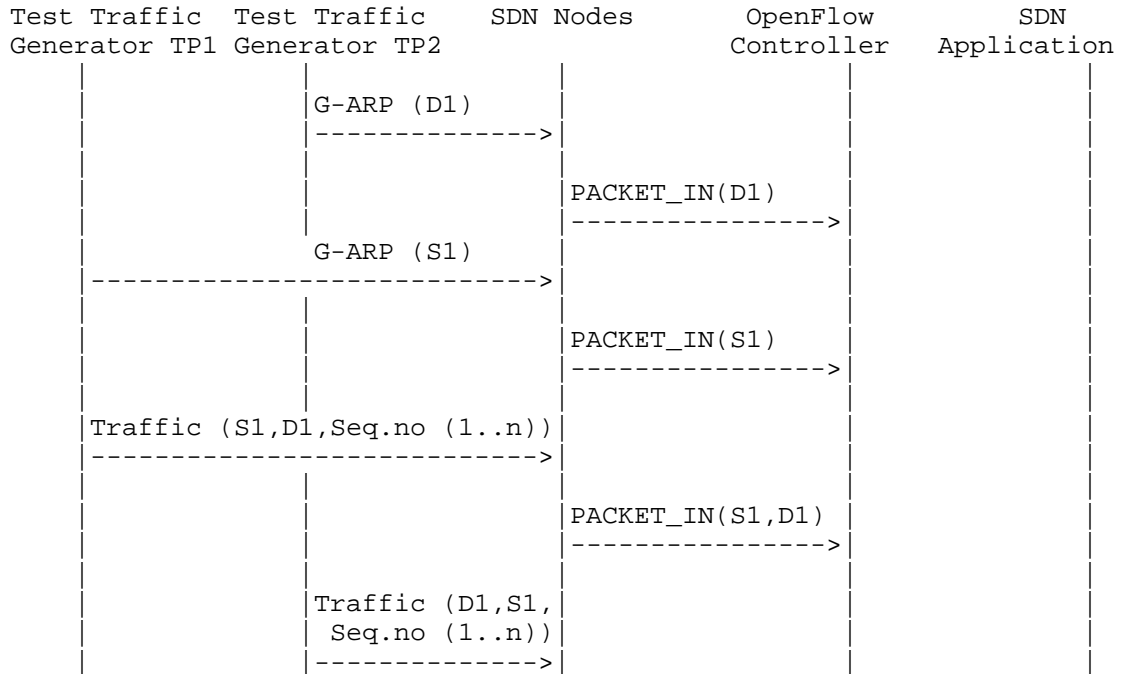
Discussion:

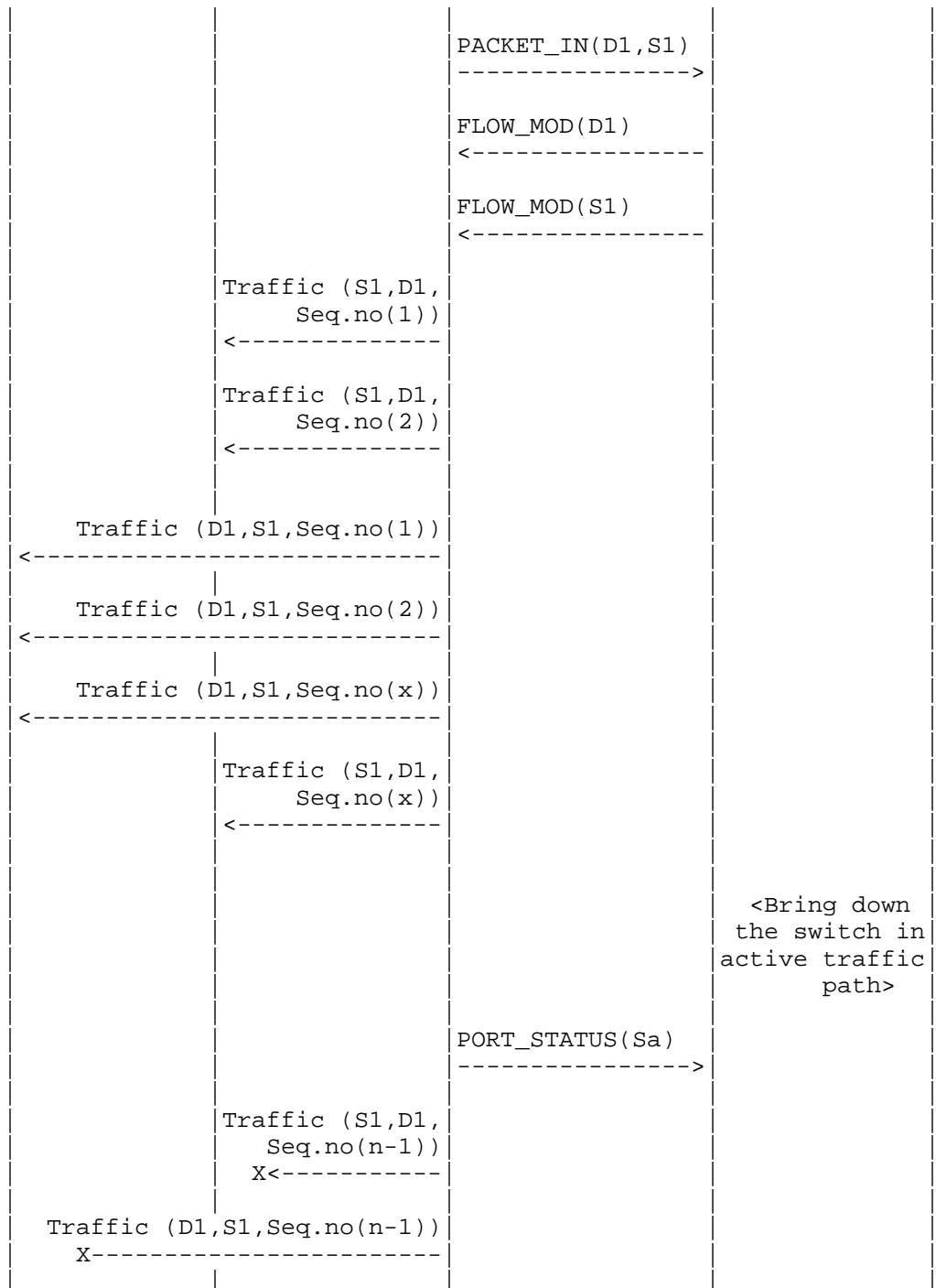
The time difference between the last valid frame received before the traffic loss and the first frame received after the traffic loss will provide the controller failover time.

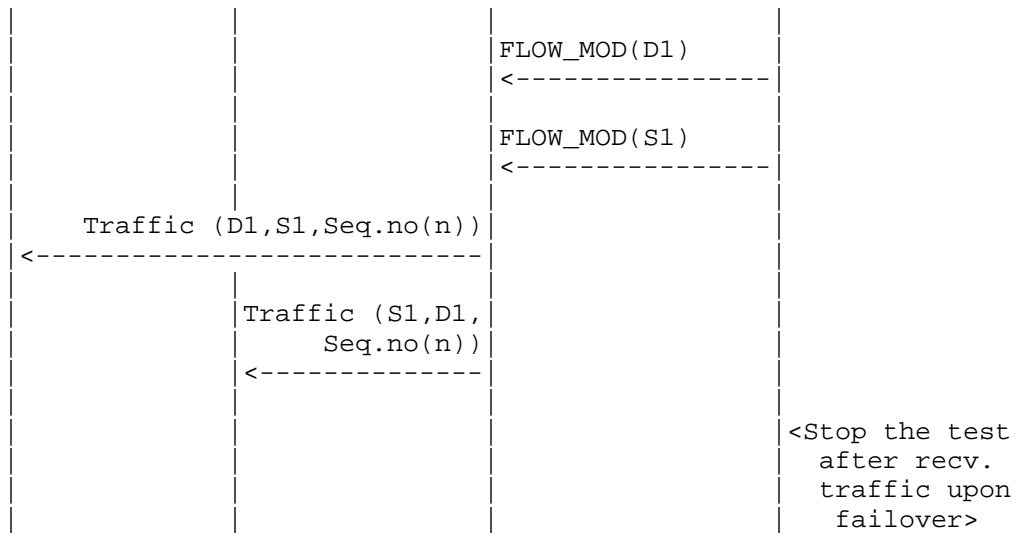
If there is no frame loss during controller failover time, the controller failover time can be deemed negligible.

11.7.2 Network Re-Provisioning Time

Procedure:





**Legend:**

G-ARP: Gratuitous ARP message.

Seq.no: Sequence number.

Sa: Neighbour switch of the switch that was brought down.

Discussion:

The time difference between the last valid frame received before the traffic loss (Packet number with sequence number x) and the first frame received after the traffic loss (packet with sequence number n) will provide the network path re-provisioning time.

Note that the test is valid only when the controller provisions the alternate path upon network failure.

12. Acknowledgements

The authors would like to thank the following individuals for providing their valuable comments to the earlier versions of this document: Al Morton (AT&T), Sandeep Gangadharan (HP), M. Georgescu (NAIST), Andrew McGregor (Google), Scott Bradner (Harvard University), Jay Karthik (Cisco), Ramakrishnan (Brocade).

13. Authors' Addresses

Bhuvaneswaran Vengainathan
Veryx Technologies Inc.
1 International Plaza, Suite 550
Philadelphia
PA 19113

Email: bhuvaneswaran.vengainathan@veryxtech.com

Anton Basil
Veryx Technologies Inc.
1 International Plaza, Suite 550
Philadelphia
PA 19113

Email: anton.basil@veryxtech.com

Mark Tassinari
Hewlett-Packard,
8000 Foothills Blvd,
Roseville, CA 95747

Email: mark.tassinari@hp.com

Vishwas Manral
Ionos Corp,
4100 Moorpark Ave,
San Jose, CA

Email: vishwas@ionosnetworks.com

Sarah Banks
VSS Monitoring
930 De Guigne Drive,
Sunnyvale, CA

Email: sbanks@encrypted.net

Internet-Draft
Network Working Group
Intended Status: Informational
Expires: March 22, 2016

Bhuvaneswaran Vengainathan
Anton Basil
Veryx Technologies
Mark Tassinari
Hewlett-Packard
Vishwas Manral
Ionos Corp
Sarah Banks
VSS Monitoring
September 23, 2015

Terminology for Benchmarking SDN Controller Performance
draft-bhuvan-bmwg-sdn-controller-benchmark-term-01

Abstract

This document defines terminology for benchmarking an SDN Controller's performance. The terms provided in this document help to benchmark SDN controller's performance independent of the controller's supported protocols and/or network services. A mechanism for benchmarking the performance of SDN controllers is defined in the companion methodology document. These two documents provide a standard mechanism to measure and evaluate the performance of various controller implementations.

Status of this Memo

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

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

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

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

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Term Definitions	4
2.1. SDN Terms	4
2.1.1. SDN Node	4
2.1.2. SDN Application.....	4
2.1.3. Flow	4
2.1.4. Northbound Interface.....	5
2.1.5. Southbound Interface.....	5
2.1.6. Controller Forwarding Table	5
2.1.7. Proactive Flow Provisioning Mode	6
2.1.8. Reactive Flow Provisioning Mode	6
2.1.9. Path	7
2.1.10. Standalone Mode.....	7
2.1.11. Cluster/Redundancy Mode.....	7
2.1.12. Asynchronous Message.....	8
2.1.13. Test Traffic Generator.....	8
2.2. Test Configuration/Setup Terms	9
2.2.1. Number of SDN Nodes.....	9
2.2.2. Test Iterations.....	9
2.2.3. Test Duration.....	9
2.2.4. Number of Cluster nodes	10
2.3. Benchmarking Terms	10
2.3.1. Performance	10
2.3.1.1. Network Topology Discovery Time	10
2.3.1.2. Asynchronous Message Processing Time.....	11
2.3.1.3. Asynchronous Message Processing Rate.....	11
2.3.1.4. Reactive Path Provisioning Time	12
2.3.1.5. Proactive Path Provisioning Time	12

2.3.1.6. Reactive Path Provisioning Rate	13
2.3.1.7. Proactive Path Provisioning Rate	13
2.3.1.8. Network Topology Change Detection Time.....	13
2.3.2. Scalability	14
2.3.2.1. Control Sessions Capacity	14
2.3.2.2. Network Discovery Size	14
2.3.2.3. Forwarding Table Capacity	15
2.3.3. Security	15
2.3.3.1. Exception Handling	15
2.3.3.2. Denial of Service Handling	16
2.3.4. Reliability	16
2.3.4.1. Controller Failover Time	16
2.3.4.2. Network Re-Provisioning Time	17
3. Test Coverage	17
4. References	18
4.1. Normative References	18
4.2. Informative References	19
5. IANA Considerations	19
6. Security Considerations	19
7. Acknowledgements	19
8. Authors' Addresses	19

1. Introduction

Software Defined Networking (SDN) is a networking architecture in which network control is decoupled from the underlying forwarding function and is placed in a centralized location called the SDN controller. The SDN controller abstracts the underlying network and offers a global view of the overall network to applications and business logic. Thus, an SDN controller provides the flexibility to program, control, and manage network behaviour dynamically through standard interfaces. Since the network controls are logically centralized, the need to benchmark the SDN controller performance becomes significant. This document defines terms to benchmark various controller designs for performance, scalability, reliability and security, independent of northbound and southbound protocols.

Conventions used in this document

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

2. Term Definitions

2.1. SDN Terms

2.1.1. SDN Node

Definition:

An SDN node is an emulated/simulated entity that forwards data in a software defined environment.

Discussion:

An SDN node can be an emulated/simulated switch, router, gateway, or any network service appliance that supports standardized or proprietary programmable interface.

Measurement Units:

N/A

See Also:

None

2.1.2. SDN Application

Definition:

Any business logic that alter the network behaviour dynamically through controller's northbound interface.

Discussion:

SDN application can be any business application, cloud orchestration system, network services orchestration etc.,

Measurement Units:

N/A

See Also:

None

2.1.3. Flow

Definition:

A flow is a uni-directional sequence of packets having common properties derived from the data contained in the packet.

Discussion:

A flow can be set of packets having same source address, destination address, source port and destination port, or any of these combinations.

Measurement Units:
N/A

See Also:
None

2.1.4. Northbound Interface

Definition:
The northbound interface is the application programming interface provided by the SDN controller for the SDN services and applications to interact with the SDN controller.

Discussion:
The northbound interface allows SDN applications and orchestration systems to program and retrieve the network information through the SDN controller.

Measurement Units:
N/A

See Also:
None

2.1.5. Southbound Interface

Definition:
The southbound interface is the application programming interface provided by the SDN controller to interact with the SDN nodes.

Discussion:
Southbound interface enables controller to interact with the SDN nodes in the infrastructure for dynamically defining the traffic forwarding behaviour.

Measurement Units:
N/A

See Also:
None

2.1.6. Controller Forwarding Table

Definition:
A controller forwarding table contains flow entries learned in one of two ways: first, entries could be learned from traffic received through the data plane, or, second, these entries could be

statically provisioned on the controller, and distributed to devices via the southbound interface.

Discussion:

The controller forwarding table has an aging mechanism which will be applied only for dynamically learnt entries.

Measurement Units:

N/A

See Also:

None

2.1.7. Proactive Flow Provisioning Mode

Definition:

Controller programming flows in SDN nodes based on the flow entries provisioned through controller's northbound interface.

Discussion:

Orchestration systems and SDN applications can define the network forwarding behaviour by programming the controller using proactive flow provisioning. The controller can then program the SDN nodes with the pre-provisioned entries.

Measurement Units:

N/A

See Also:

None

2.1.8. Reactive Flow Provisioning Mode

Definition:

Controller programming flows in SDN nodes based on the traffic received from SDN nodes through controller's southbound interface

Discussion:

The SDN controller dynamically decides the forwarding behaviour based on the incoming traffic from the SDN nodes. The controller then programs the SDN nodes using Reactive Flow Provisioning.

Measurement Units:

N/A

See Also:

None

2.1.9. Path

Definition:

A path is a sequence of SDN nodes and links traversed by a flow.

Discussion:

As defined in RFC 2330, path is a sequence of the form $\langle h_0, l_1, h_1, \dots, l_n, h_n \rangle$, where $n \geq 0$, h_0 and h_n is a Host, $h_1 \dots h_{n-1}$ is an SDN Node, each l_i is a link between h_{i-1} and h_i . A pair $\langle l_i, h_i \rangle$ is termed a 'hop'. Note that path is a unidirectional concept.

Measurement Units:

N/A

See Also:

None

2.1.10. Standalone Mode

Definition:

Single controller handling all control plane functionalities without redundancy, or the ability to provide high availability and/or automatic failover.

Discussion:

In standalone mode, one controller manages one or more network domains.

Measurement Units:

N/A

See Also:

None

2.1.11. Cluster/Redundancy Mode

Definition:

A group of 2 or more controllers handling all control plane functionalities.

Discussion:

In cluster mode, multiple controllers are teamed together for the purpose of load sharing and/or high availability. The controllers in

the group may work in active/standby (master/slave) or active/active (equal) mode depending on the intended purpose.

Measurement Units:
N/A

See Also:
None

2.1.12. Asynchronous Message

Definition:
Any message from the SDN node that is generated for network events.

Discussion:
Control messages like flow setup request and response message is classified as asynchronous message. The controller has to return a response message. Note that the SDN node will not be in blocking mode and continues to send/receive other control messages

Measurement Units:
N/A

See Also:
None

2.1.13. Test Traffic Generator

Definition:
Test Traffic Generator is an entity that generates/receives network traffic.

Discussion:
Test Traffic Generator can be an entity that interfaces with SDN Nodes to send/receive real-time network traffic.

Measurement Units:
N/A

See Also:
None

2.2. Test Configuration/Setup Terms

2.2.1. Number of SDN Nodes

Definition:

The number of SDN nodes present in the defined test topology.

Discussion:

The SDN nodes defined in the test topology can be deployed using real hardware or emulated in hardware platforms.

Measurement Units:

N/A

See Also:

None

2.2.2. Test Iterations

Definition:

The number of times the test needs to be repeated.

Discussion:

The test needs to be repeated for multiple iterations to obtain a reliable metric. It is recommended that this test SHOULD be performed for at least 10 iterations to increase the confidence in measured result.

Measurement Units:

N/A

See Also:

None

2.2.3. Test Duration

Definition:

Defines the duration of test trails for each iteration.

Discussion:

Test duration forms the basis for stop criteria for benchmarking tests. Test not completed within this time interval is considered as incomplete.

Measurement Units:
seconds

See Also:
None

2.2.4. Number of Cluster nodes

Definition:

Defines the number of controllers present in the controller cluster.

Discussion:

This parameter is relevant when testing the controller performance in clustering/teaming mode. The number of nodes in the cluster MUST be greater than 1.

Measurement Units:
N/A

See Also:
None

2.3. Benchmarking Terms

This section defines metrics for benchmarking the SDN controller. The procedure to perform the defined metrics is defined in the accompanying methodology document.

2.3.1. Performance

2.3.1.1. Network Topology Discovery Time

Definition:

To measure the time taken to discover the network topology - nodes and links by a controller.

Discussion:

Network topology discovery is key for the SDN controller to provision and manage the network. So it is important to measure how quickly the controller discovers the topology to learn the current network state. This benchmark is obtained by presenting a network topology (Tree, Mesh or Linear) with the given number of nodes to the controller and wait for the discovery process to complete. It is expected that the controller supports network discovery mechanism and uses protocol messages for its discovery process.

Measurement Units:
milliseconds

See Also:
None

2.3.1.2. Asynchronous Message Processing Time

Definition:

To measure the time taken by the controller to process an asynchronous message.

Discussion:

For SDN to support dynamic network provisioning, it is important to measure how quickly the controller responds to an event triggered from the network. The event could be any notification messages generated by an SDN node upon arrival of a new flow, link down etc. This benchmark is obtained by sending asynchronous messages from every connected SDN nodes one at a time for the defined test duration. This test assumes that the controller will respond to the received asynchronous message.

Measurement Units:
milliseconds

See Also:
None

2.3.1.3. Asynchronous Message Processing Rate

Definition:

To measure the maximum number of asynchronous messages that a controller can process within the test duration.

Discussion:

As SDN assures flexible network and agile provisioning, it is important to measure how many network events that the controller can handle at a time. This benchmark is obtained by sending asynchronous messages from every connected SDN nodes at full connection capacity for the given test duration. This test assumes that the controller will respond to all the received asynchronous messages.

Measurement Units:
Messages processed per second.

See Also:
None

2.3.1.4. Reactive Path Provisioning Time

Definition:

The time taken by the controller to setup a path reactively between source and destination node, expressed in milliseconds.

Discussion:

As SDN supports agile provisioning, it is important to measure how fast that the controller provisions an end-to-end flow in the dataplane. The benchmark is obtained by sending traffic from a source endpoint to the destination endpoint, finding the time difference between the first and the last flow provisioning message exchanged between the controller and the SDN nodes for the traffic path.

Measurement Units:
milliseconds.

See Also:
None

2.3.1.5. Proactive Path Provisioning Time

Definition:

The time taken by the controller to setup a path proactively between source and destination node, expressed in milliseconds.

Discussion:

For SDN to support pre-provisioning of traffic path from application, it is important to measure how fast that the controller provisions an end-to-end flow in the dataplane. The benchmark is obtained by provisioning a flow on controller's northbound interface for the traffic to reach from a source to a destination endpoint, finding the time difference between the first and the last flow provisioning message exchanged between the controller and the SDN nodes for the traffic path.

Measurement Units:
milliseconds.

See Also:
None

2.3.1.6. Reactive Path Provisioning Rate

Definition:

Measure the maximum number of independent paths a controller can concurrently establish between source and destination nodes reactively within the test duration, expressed in paths per second.

Discussion:

For SDN to support agile traffic forwarding, it is important to measure how many end-to-end flows that the controller could setup in the dataplane. This benchmark is obtained by sending traffic each with unique source and destination pairs from the source SDN node and determine the number of frames received at the destination SDN node.

Measurement Units:

Paths provisioned per second.

See Also:

None

2.3.1.7. Proactive Path Provisioning Rate

Definition:

Measure the maximum number of independent paths a controller can concurrently establish between source and destination nodes proactively within the test duration, expressed in paths per second.

Discussion:

For SDN to support pre-provisioning of traffic path for a larger network from the application, it is important to measure how many end-to-end flows that the controller could setup in the dataplane. This benchmark is obtained by sending traffic each with unique source and destination pairs from the source SDN node. Program the flows on controller's northbound interface for traffic to reach from each of the unique source and destination pairs and determine the number of frames received at the destination SDN node.

Measurement Units:

Paths provisioned per second.

See Also:

None

2.3.1.8. Network Topology Change Detection Time

Definition:

The amount of time required for the controller to detect any changes in the network topology.

Discussion:

In order to for the controller to support fast network failure recovery, it is critical to measure how fast the controller is able to detect any network-state change events. This benchmark is obtained by triggering a topology change event and measuring the time controller takes to detect and initiate a topology re-discovery process.

Measurement Units:
milliseconds

See Also:
None

2.3.2. Scalability

2.3.2.1. Control Sessions Capacity

Definition:

To measure the maximum number of control sessions the controller can maintain.

Discussion:

Measuring the controller's control sessions capacity is important to determine the controller's system and bandwidth resource requirements. This benchmark is obtained by establishing control session with the controller from each of the SDN node until it fails. The number of sessions that were successfully established will provide the Control Sessions Capacity.

Measurement Units:
N/A

See Also:
None

2.3.2.2. Network Discovery Size

Definition:

To measure the network size (number of nodes, links and hosts) that a controller can discover.

Discussion:

For optimal network planning, it is key to measure the maximum network size that the controller can discover. This benchmark is obtained by presenting an initial set of SDN nodes for discovery to the controller. Based on the initial discovery, the number of SDN nodes is increased or decreased to determine the maximum nodes that the controller can discover.

Measurement Units:

N/A

See Also:

None

2.3.2.3. Forwarding Table Capacity

Definition:

The maximum number of flow entries that a controller can manage in its Forwarding table.

Discussion:

It is significant to measure the capacity of controller's Forwarding Table to determine the number of flows that controller could forward without flooding/dropping. This benchmark is obtained by continuously presenting the controller with new flow entries through reactive or proactive flow provisioning mode until the forwarding table becomes full. The maximum number of nodes that the controller can hold in its Forwarding Table will provide Forwarding Table Capacity.

Measurement Units:

Maximum number of flow entries managed.

See Also:

None

2.3.3. Security

2.3.3.1. Exception Handling

Definition:

To determine the effect of handling error packets and notifications on performance tests.

Discussion:

This benchmark test is to be performed after obtaining the baseline performance of the performance tests defined in Section 2.3.1. This

benchmark determines the deviation from the baseline performance due to the handling of error or failure messages from the connected SDN nodes.

Measurement Units:

N/A

See Also:

None

2.3.3.2. Denial of Service Handling

Definition:

To determine the effect of handling denial of service (DoS) attacks on performance and scalability tests.

Discussion:

This benchmark test is to be performed after obtaining the baseline performance of the performance and scalability tests defined in section 2.3.1 and section 2.3.1.. This benchmark determines the deviation from the baseline performance due to the handling of denial of service attacks on controller.

Measurement Units:

Deviation of baseline metrics while handling Denial of Service Attacks.

See Also:

None

2.3.4. Reliability

2.3.4.1. Controller Failover Time

Definition:

The time taken to switch from an active controller to the backup controller, when the controllers work in redundancy mode and the active controller fails.

Discussion:

This benchmark determine the impact of provisioning new flows when controllers are teamed and the active controller fails.

Measurement Units:

milliseconds.

See Also:

None

2.3.4.2. Network Re-Provisioning Time

Definition:

The time taken to re-route the traffic by the Controller, when there is a failure in existing traffic paths.

Discussion:

This benchmark determines the controller's re-provisioning ability upon network failures. This benchmark test assumes the following:

- i. Network topology supports redundant path between source and destination endpoints.
- ii. Controller does not pre-provision the redundant path.

Measurement Units:

milliseconds.

See Also:

None

3. Test Coverage

	Speed	Scalability	Reliability
Setup	1. Network Topology Discovery	1. Network Discovery Size	
	2. Reactive Path Provisioning Time		
	3. Proactive Path Provisioning Time		
	4. Reactive Path Provisioning Rate		
	5. Proactive Path Provisioning Rate		

Operational	<ol style="list-style-type: none"> 1. Asynchronous Message Processing Rate 2. Asynchronous Message Processing Time 	<ol style="list-style-type: none"> 1. Control Sessions Capacity 2. Forwarding Table Capacity 	<ol style="list-style-type: none"> 1. Network Topology Change Detection Time 2. Exception Handling 3. Denial of Service Handling 4. Network Re-Provisioning Time
Tear Down			<ol style="list-style-type: none"> 1. Controller Failover Time

4. References

4.1. Normative References

- [RFC2330] V. Paxson, G. Almes, J. Mahdavi, M. Mathis, "Framework for IP Performance Metrics", RFC 2330, May 1998.
- [RFC6241] R. Enns, M. Bjorklund, J. Schoenwaelder, A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC6020] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010
- [RFC5440] JP. Vasseur, JL. Le Roux, "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, March 2009.
- [OpenFlow Switch Specification] ONF, "OpenFlow Switch Specification" Version 1.4.0 (Wire Protocol 0x05), October 14, 2013.
- [I-D.sdn-controller-benchmark-meth] Bhuvaneshwaran.V, Anton Basil,

Mark.T, Vishwas Manral, Sarah Banks "Benchmarking
Methodology for SDN Controller Performance",
draft-bhuvan-bmwg-sdn-controller-benchmark-meth-01
(Work in progress), July 19, 2015

4.2. Informative References

[OpenContrail] Ankur Singla, Bruno Rijsman, "OpenContrail
Architecture Documentation",
<http://opencontrail.org/opencontrail-architecture-documentation>

[OpenDaylight] OpenDaylight Controller:Architectural Framework,
https://wiki.opendaylight.org/view/OpenDaylight_Controller

5. IANA Considerations

This document does not have any IANA requests.

6. Security Considerations

Security issues are not discussed in this memo.

7. Acknowledgements

The authors would like to acknowledge Sandeep Gangadharan (HP) for the significant contributions to the earlier versions of this document. The authors would like to thank the following individuals for providing their valuable comments to the earlier versions of this document: Al Morton (AT&T), M. Georgescu (NAIST), Andrew McGregor (Google), Scott Bradner (Harvard University), Jay Karthik (Cisco), Ramakrishnan (Brocade), Khasanov Boris (Huawei).

8. Authors' Addresses

Bhuvaneswaran Vengainathan
Veryx Technologies Inc.
1 International Plaza, Suite 550
Philadelphia
PA 19113

Email: bhuvaneswaran.vengainathan@veryxtech.com

Anton Basil
Veryx Technologies Inc.
1 International Plaza, Suite 550

Philadelphia
PA 19113

Email: anton.basil@veryxtech.com

Mark Tassinari
Hewlett-Packard,
8000 Foothills Blvd,
Roseville, CA 95747

Email: mark.tassinari@hp.com

Vishwas Manral
Ionos Corp,
4100 Moorpark Ave,
San Jose, CA

Email: vishwas@ionosnetworks.com

Sarah Banks
VSS Monitoring

Email: sbanks@encrypted.net

SDN Research Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

LM. Contreras
Telefonica
CJ. Bernardos
UC3M
D. Lopez
Telefonica
M. Boucadair
France Telecom
P. Iovanna
Ericsson
October 19, 2015

Cooperating Layered Architecture for SDN
draft-contreras-sdnrg-layered-sdn-04

Abstract

Software Defined Networking proposes the separation of the control plane from the data plane in the network nodes and its logical centralization on a control entity. Most of the network intelligence is moved to this functional entity. Typically, such entity is seen as a compendium of interacting control functions in a vertical, tight integrated fashion. The relocation of the control functions from a number of distributed network nodes to a logical central entity conceptually places together a number of control capabilities with different purposes. As a consequence, the existing solutions do not provide a clear separation between transport control and services that relies upon transport capabilities.

This document describes a proposal named Cooperating Layered Architecture for SDN. The idea behind that is to differentiate the control functions associated to transport from those related to services, in such a way that they can be provided and maintained independently, and can follow their own evolution path.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 21, 2016.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. Terminology	4
3. Architecture overview	5
3.1. Functional strata	8
3.1.1. Transport stratum	8
3.1.2. Service stratum	9
3.1.3. Recursiveness	9
3.2. Plane separation	9
3.2.1. Control Plane	9
3.2.2. Management Plane	10
3.2.3. Resource Plane	10
4. Required features	10
5. Communication between SDN Controllers	11
6. Deployment scenarios	11
6.1. Full SDN environments	11
6.1.1. Multiple Service strata associated to a single Transport stratum	11
6.1.2. Single service stratum associated to multiple Transport strata	12
6.2. Hybrid environments	12
6.2.1. SDN Service stratum associated to a legacy Transport stratum	12
6.2.2. Legacy Service stratum associated to an SDN Transport stratum	12
6.3. Multi-domain scenarios in Transport Stratum	12
7. Use cases	13

7.1. Network Function Virtualization	13
7.2. Abstraction and Control of Transport Networks	13
8. IANA Considerations	13
9. Security Considerations	13
10. References	13
10.1. Normative References	13
10.2. Informative References	13
Authors' Addresses	14

1. Introduction

Software Defined Networking (SDN) proposes the separation of the control plane from the data plane in the network nodes and its logical centralization on a control entity. A programmatic interface is defined between such entity and the network nodes, which functionality is supposed to perform traffic forwarding (only). Through that interface, the control entity instructs the nodes involved in the forwarding plane and modifies their traffic forwarding behavior accordingly.

Most of the intelligence is moved to such functional entity. Typically, such entity is seen as a compendium of interacting control functions in a vertical, tight integrated fashion.

This approach presents a number of issues:

- o Unclear responsibilities between actors involved in a service provision and delivery.
- o Complex reuse of functions for the provision of services.
- o Closed, monolithic control architectures.
- o Difficult interoperability and interchangeability of functional components.
- o Blurred business boundaries among providers.
- o Complex service/network diagnosis and troubleshooting, particularly to determine which segment is responsible for a failure.

The relocation of the control functions from a number of distributed network nodes to another entity conceptually places together a number of control capabilities with different purposes. As a consequence, the existing solutions do not provide a clear separation between services and transport control.

This document describes a proposal named Cooperating Layered Architecture for SDN (CLAS). The idea behind that is to differentiate the control functions associated to transport from those related to services, in such a way that they can be provided and maintained independently, and can follow their own evolution path.

Despite such differentiation it is required a close cooperation between service and transport layers and associated components to provide an efficient usage of the resources.

2. Terminology

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

This document makes use of the following terms:

- o Transport: denotes the transfer capabilities offered by a networking infrastructure. The transfer capabilities can rely upon pure IP techniques, or other means such as MPLS or optics.
- o Service: denote a logical construct that make use of transport capabilities. This document does not make any assumption on the functional perimeter of a service that can be built above a transport infrastructure. As such, a service can be an offering that is offered to customers or be invoked for the delivery of another (added-value) service.
- o SDN intelligence: refers to the decision-making process that is hosted by a node or a set of nodes. The intelligence can be centralized or distributed. Both schemes are within the scope of this document. The SDN intelligence relies on inputs from various functional blocks such as: network topology discovery, service topology discovery, resource allocation, business guidelines, customer profiles, service profiles, etc. The exact decomposition of an SDN intelligence, apart from the layering discussed in this document, is out of scope.

Additionally, the following acronyms are used in this document.

CLAS: Cooperating Layered Architecture for SDN

FCAPS: Fault, Configuration, Accounting, Performance and Security

SDN: Software Defined Networking

SLA: Service Level Agreement

3. Architecture overview

Current operator networks support multiple services (e.g., VoIP, IPTV, mobile VoIP, critical mission applications, etc.) on a variety of transport technologies. The provision and delivery of a service independently of the underlying transport capabilities requires a separation of the service related functionalities and an abstraction of the transport network to hide the specificities of underlying transfer techniques while offering a common set of capabilities.

Such separation can provide configuration flexibility and adaptability from the point of view of either the services or the transport network. Multiple services can be provided on top of a common transport infrastructure, and similarly, different technologies can accommodate the connectivity requirements of a certain service. A close coordination among them is required for a consistent service delivery (inter-layer cooperation).

This document focuses particularly on:

- o Means to expose transport capabilities to external services.
- o Means to capture service requirements of services.
- o Means to notify service intelligence with underlying transport events, for example to adjust service decision-making process with underlying transport events.
- o Means to instruct the underlying transport capabilities to accommodate new requirements, etc.

An example is to guarantee some Quality of Service (QoS) levels. Different QoS-based offerings could be present at both service and transport layers. Vertical mechanisms for linking both service and transport QoS mechanisms should be in place to provide the quality guarantees to the end user.

CLAS architecture assumes that the logically centralized control functions are separated in two functional blocks or layers. One of the functional blocks comprises the service-related functions, whereas the other one contains the transport-related functions. The cooperation between the two layers is considered to be implemented through standard interfaces.

Figure 1 shows the CLAS architecture. It is based on functional separation in the NGN architecture defined by the ITU-T in [Y.2011].

Two strata of functionality are defined, namely the Service Stratum, comprising the service-related functions, and the Transport Stratum, covering the transport ones. The functions on each of these layers are further grouped on control, management and user (or data) planes.

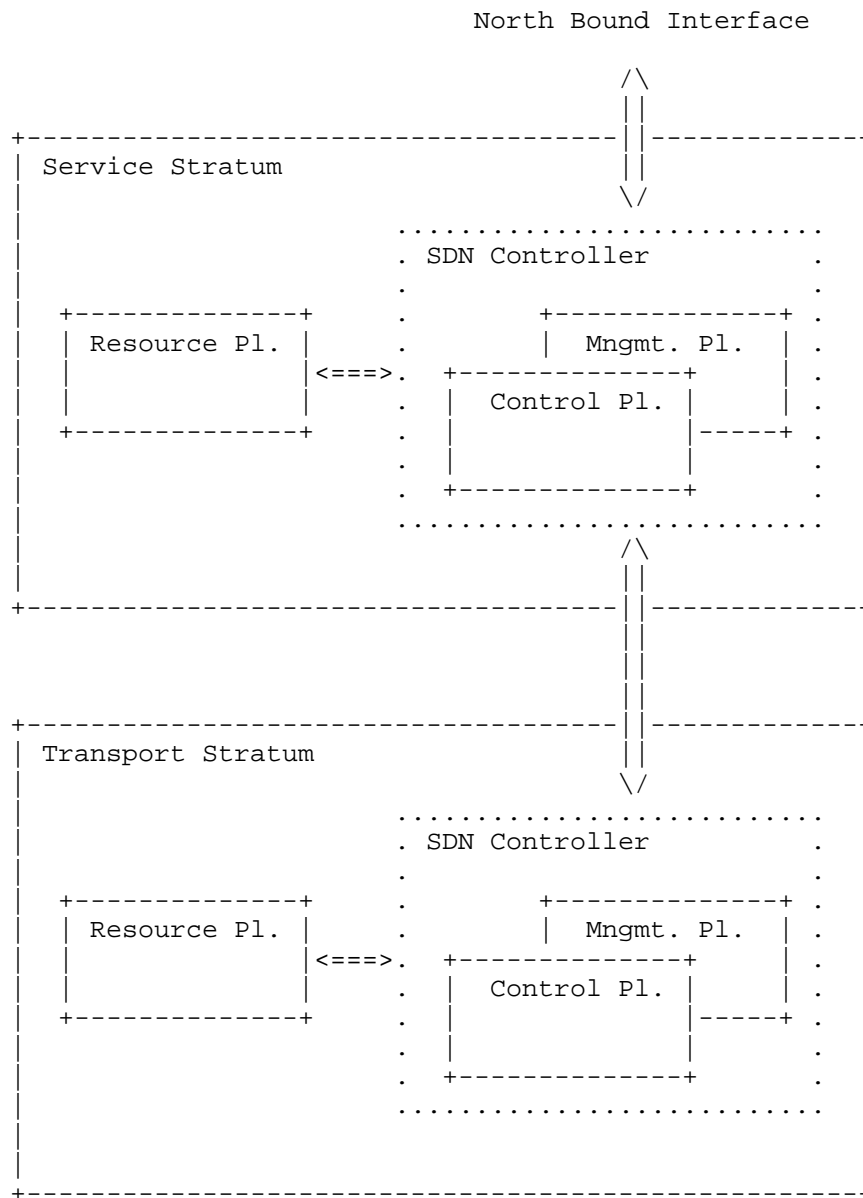


Figure 1: Cooperating Layered Architecture for SDN

In the CLAS architecture both the control and management functions are the ones logically centralized in one or a set of SDN controllers, in such a way that separated SDN controllers are present

in the Service and Transport strata. Furthermore, the generic user or data plane functions included in the NGN architecture are referred here as resource plane functions. The resource plane in each stratum is controlled by the corresponding SDN controller through a standard interface.

The SDN controllers cooperate for the provision and delivery of services. There is a hierarchy in which the Service SDN controller requests transport capabilities to the Transport SDN controller. Furthermore, the Transport SDN controller interacts with the Service SDN controller to inform it about events in the transport network that can motivate actions in the service layer.

The Service SDN controller acts as a client of the Transport SDN controller.

Despite it is not shown in the figure, the Resource planes of each stratum could be connected. This will depend on the kind of service provided. Furthermore, the Service stratum could offer an interface towards external applications to expose network service capabilities to those applications or customers.

This document does assume that SDN techniques can be enabled jointly with other distributed means (e.g., IGP).

3.1. Functional strata

As described before, the functional split separates transport-related functions from service-related functions. Both strata cooperate for a consistent service delivery.

Consistency is determined and characterized by the service layer.

Communication between these two components could be implemented using a variety of means (such as [I-D.boucadair-connectivity-provisioning-protocol], Intermediate-Controller Plane Interface (I-CPI) [ONFArch], etc).

3.1.1. Transport stratum

The Transport stratum comprises the functions focused on the transfer of data between the communication end points (e.g., between end-user devices, between two service gateways, etc.). The data forwarding nodes are controlled and managed by the Transport SDN component. The Control plane in the SDN controller is in charge of instructing the forwarding devices to build the end to end data path for each communication or to make sure forwarding service is appropriately setup. Forwarding may not be rely on the sole pre-configured

entries; dynamic means can be enabled so that involved nodes can build dynamically routing and forwarding paths. Finally, the Management plane performs management functions (i.e., FCAPS) on those devices, like fault or performance management, as part of the Transport stratum capabilities.

3.1.2. Service stratum

The Service stratum contains the functions related to the provision of services and the capabilities offered to external applications. The Resource plane consists of the resources involved in the service delivery, such as computing resources, registries, databases, etc. The Control plane is in charge of controlling and configuring those resources, as well as interacting with the Control plane of the Transport stratum in client mode for requesting transport capabilities for a given service. In the same way, the Management plane implements management actions on the service-related resources and interacts with the Management plane in the Transport stratum for a cooperating management between layers.

3.1.3. Recursiveness

Recursive layering can happen in some usage scenarios in which the Transport Stratum is itself structured in Service and Transport Stratum. This could be the case of the provision of a transport services complemented with advanced capabilities additional to the pure data transport (e.g., maintenance of a given SLA [RFC7297]).

3.2. Plane separation

The CLAS architecture leverages on the SDN proposition of plane separation. As mentioned before, three different planes are considered for each stratum. The communication among these three planes (and with the corresponding plane in other strata) is based on open, standard interfaces.

3.2.1. Control Plane

The Control plane logically centralizes the control functions of each stratum and directly controls the corresponding resources. [RFC7426] introduces the role of the control plane in a SDN architecture. This plane is part of an SDN controller, and can interact with other control planes in the same or different strata for accomplishing control functions.

3.2.2. Management Plane

The Management plane logically centralizes the management functions for each stratum, including the management of the Control and Resource planes. [RFC7426] describes the functions of the management plane in a SDN environment. This plane is also part of the SDN controller, and can interact with the corresponding management planes residing in SDN controllers of the same or different strata.

3.2.3. Resource Plane

The Resource plane comprises the resources for either the transport or the service functions. In some cases the service resources can be connected to the transport ones (e.g., being the terminating points of a transport function) whereas in other cases it can be decoupled from the transport resources (e.g., one database keeping some register for the end user). Both forwarding and operational planes proposed in [RFC7426] would be part of the Resource plane in this architecture.

4. Required features

A number of features are required to be supported by the CLAS architecture.

- o Abstraction: the mapping of physical resources into the corresponding abstracted resources.
- o Service parameter translation: translation of service parameters (e.g., in the form of SLAs) to transport parameters (or capabilities) according to different policies.
- o Monitoring: mechanisms (e.g. event notifications) available in order to dynamically update the (abstracted) resources' status taking in to account e.g. the traffic load.
- o Resource computation: functions able to decide which resources will be used for a given service request. As an example, functions like PCE could be used to compute/select/decide a certain path.
- o Orchestration: ability to combine diverse resources (e.g., IT and network resources) in an optimal way.
- o Accounting: record of resource usage.
- o Security: secure communication among components, preventing e.g. DoS attacks.

5. Communication between SDN Controllers

The SDN Controller residing respectively in the Service and the Transport Stratum need to establish a tight coordination. Mechanisms for transfer relevant information for each stratum should be defined.

From the Service perspective, the Service SDN controller needs to easily access transport resources through well defined APIs to access the capabilities offered by the Transport Stratum. There could be different ways of obtainign such transport-aware information, i.e., by discovering or publishing mechanisms. In the former case the Service SDN Controller could be able of handling complete information about the transport capabilities (including resources) offered by the Transport Stratum. In the latter case, the Transport Stratum exposes available capabilities e.g. through a catalog, reducing the amount of detail of the underlying network.

On the other hand, the Transport Stratum requires to properly capture Service requirements. These can include SLA requirements with specific metrics (such as delay), level of protection to be provided, max/min capacity, applicable resource constraints, etc.

The communication between controllers should be also secure, preventing denial of service.

6. Deployment scenarios

Different situations can be found depending on the characteristics of the networks involved in a given deployment.

6.1. Full SDN environments

This case considers the fact that the networks involved in the provision and delivery of a given service have SDN capabilities.

6.1.1. Multiple Service strata associated to a single Transport stratum

A single Transport stratum can provide transfer functions to more than one Service strata. The Transport stratum offers a standard interface to each of the Service strata. The Service strata are the clients of the Transport stratum. Some of the capabilities offered by the Transport stratum can be isolation of the transport resources (slicing), independent routing, etc.

6.1.2. Single service stratum associated to multiple Transport strata

A single Service stratum can make use of different Transport strata for the provision of a certain service. The Service stratum interfaces each of the Transport strata with standard protocols, and orchestrates the provided transfer capabilities for building the end to end transport needs.

6.2. Hybrid environments

This case considers scenarios where one of the strata is legacy totally or in part.

6.2.1. SDN Service stratum associated to a legacy Transport stratum

An SDN service stratum can interact with a legacy Transport stratum through some interworking function able to adapt SDN-based control and management service-related commands to legacy transport-related protocols, as expected by the legacy Transport stratum. The SDN controller in the Service stratum is not aware of the legacy nature of the underlying Transport stratum.

6.2.2. Legacy Service stratum associated to an SDN Transport stratum

A legacy Service stratum can work with an SDN-enabled Transport stratum through the mediation of an interworking function capable to interpret commands from the legacy service functions and translate them into SDN protocols for operating with the SDN-enabled Transport stratum.

6.3. Multi-domain scenarios in Transport Stratum

The Transport Stratum can be composed by transport resources being part of different administrative, topological or technological domains. The Service Stratum can yet interact with a single entity in the Transport Stratum in case some abstraction capabilities are provided in the transport part to emulate a single stratum.

Those abstraction capabilities constitute a service itself offered by the Transport Stratum to the services making use of it. This service is focused on the provision of transport capabilities, then different of the final communication service using such capabilities.

In this particular case this recursion allows multi-domain scenarios at transport level.

Multi-domain situations can happen in both single-operator and multi-operator scenarios. Multi-operator scenarios will be addressed in future versions of the document.

In single operator scenarios a multi-domain or end-to-end abstraction component can provide an homogeneous abstract view of the underlying heterogeneous transport capabilities for all the domains.

7. Use cases

This section presents a number of use cases as examples of applicability of this proposal

7.1. Network Function Virtualization

To be completed

7.2. Abstraction and Control of Transport Networks

To be completed.

8. IANA Considerations

TBD.

9. Security Considerations

TBD. Security in the communication between strata to be addressed.

10. References

10.1. Normative References

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

[Y.2011] "General principles and general reference model for Next Generation Networks", ITU-T Recommendation Y.2011 , October 2004.

10.2. Informative References

- [I-D.boucadair-connectivity-provisioning-protocol]
Boucadair, M., Jacquenet, C., Zhang, D., and P. Georgatsos, "Connectivity Provisioning Negotiation Protocol (CPNP)", draft-boucadair-connectivity-provisioning-protocol-10 (work in progress), September 2015.
- [ONFArch] Open Networking Foundation, "SDN Architecture, Issue 1", June 2014,
<https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf>.
- [RFC7297] Boucadair, M., Jacquenet, C., and N. Wang, "IP Connectivity Provisioning Profile (CPP)", RFC 7297, DOI 10.17487/RFC7297, July 2014,
<<http://www.rfc-editor.org/info/rfc7297>>.
- [RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <<http://www.rfc-editor.org/info/rfc7426>>.

Authors' Addresses

Luis M. Contreras
Telefonica
Ronda de la Comunicacion, s/n
Sur-3 building, 3rd floor
Madrid 28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com
URI: <http://people.tid.es/LuisM.Contreras/>

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Phone: +34 91624 6236
Email: cjbc@it.uc3m.es
URI: <http://www.it.uc3m.es/cjbc/>

Diego R. Lopez
Telefonica
Ronda de la Comunicacion, s/n
Sur-3 building, 3rd floor
Madrid 28050
Spain

Email: diego.r.lopez@telefonica.com

Mohamed Boucadair
France Telecom
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Paola Iovanna
Ericsson
Pisa
Italy

Email: paola.iovanna@ericsson.com

PCE Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2016

Quintin Zhao
Robin Li
Huawei Technologies
King Ke
Tencent Holdings Ltd.
Luyuan Fang
Microsoft
Chao Zhou
Cisco Systems
Boris Zhang
Telus Communications
July 6, 2015

The Use Cases for Using PCE as the Central Controller(PCECC) of LSPs
draft-zhao-pce-central-controller-user-cases-02

Abstract

In certain networks deployment scenarios, service providers would like to keep all the existing MPLS functionalities in both MPLS and GMPLS network while removing the complexity of existing signaling protocols such as LDP and RSVP-TE. In this document, we propose to use the PCE as a central controller so that LSP can be calculated/signaled/initiated/downloaded/managed through a centralized PCE server to each network devices along the LSP path while leveraging the existing PCE technologies as much as possible.

This draft describes the use cases for using the PCE as the central controller where LSPs are calculated/setup/initiated/downloaded/maintained through extending the current PCE architectures and extending the PCEP.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 7, 2016.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
1.1. Background	3
1.2. Using the PCE as the Central Controller (PCECC) Approach	4
2. Terminology	7
3. PCEP Requirements	7
4. Use Cases of PCECC for Label Resource Reservations	8
5. Using PCECC for SR without the IGP Extension	9
5.1. Use Cases of PCECC for SR Best Effort(BE) Path	10
5.2. Use Cases of PCECC for SR Traffic Engineering (TE) Path	11
6. Use Cases of PCECC for TE LSP	12
7. Use Cases of PCECC for Multicast LSPs	14
7.1. Using PCECC for P2MP/MP2MP LSPs' Setup	14
7.2. Use Cases of PCECC for the Resiliency of P2MP/MP2MP LSPs	15
7.2.1. PCECC for the End-to-End Protection of the P2MP/MP2MP LSPs	15
7.2.2. PCECC for the Local Protection of the P2MP/MP2MP LSPs	16
8. Use Cases of PCECC for LSP in the Network Migration	17
9. Use Cases of PCECC for L3VPN and PWE3	19
10. The Considerations for PCECC Procedure and PCEP extensions	19
11. IANA Considerations	20
12. Security Considerations	20
13. Acknowledgments	20
14. References	20
14.1. Normative References	20
14.2. Informative References	20

1. Introduction

1.1. Background

In certain network deployment scenarios, service providers would like to have the ability to dynamically adapt to a wide range of customer's requests for the sake of flexible network service delivery, SDN has provides additional flexibility in how the network is operated comparing the traditional network.

The existing networking ecosystem has become awfully complex and highly demanding in terms of robustness, performance, scalability, flexibility, agility, etc. By migrating to the SDN enabled network from the existing network, service providers and network operators must have a solution which they can evolve easily from the existing network into the SDN enabled network while keeping the network services remain scalable, guarantee robustness and availability etc.

Taking the smooth transition between traditional network and the new SDN enabled network into account, especially from a cost impact assessment perspective, using the existing PCE components from the current network to function as the central controller of the SDN network is one choice, which not only achieves the goal of having a centralized controller to provide the functionalities needed for the central controller, but also leverages the existing PCE network components.

The Path Computation Element communication Protocol (PCEP) provides mechanisms for Path Computation Elements (PCEs) to perform route computations in response to Path Computation Clients (PCCs) requests. PCEP Extensions for PCE-initiated LSP Setup in a Stateful PCE Model draft [I-D. draft-ietf-pce- stateful-pce] describes a set of extensions to PCEP to enable active control of MPLS-TE and GMPLS tunnels.

[I-D.crabbe-pce-pce-initiated-lsp] describes the setup and teardown of PCE-initiated LSPs under the active stateful PCE model, without the need for local configuration on the PCC, thus allowing for a dynamic MPLS network that is centrally controlled and deployed.

[I-D.ali-pce-remote-initiated-gmpls-lsp] complements [I-D. draft-crabbe-pce-pce-initiated-lsp] by addressing the requirements for remote-initiated GMPLS LSPs.

SR technology leverages the source routing and tunneling paradigms. A source node can choose a path without relying on hop-by-hop signaling protocols such as LDP or RSVP-TE. Each path is specified as a set of "segments" advertised by link-state routing protocols

(IS-IS or OSPF). [I-D.filsfils-spring-segment-routing] provides an introduction to SR technology. The corresponding IS-IS and OSPF extensions are specified in [I-D.ietf-isis-segment-routing-extensions] and [I-D.psenak-ospf-segment-routing-extensions], respectively.

A Segment Routed path (SR path) can be derived from an IGP Shortest Path Tree (SPT). Segment Routed Traffic Engineering paths (SR-TE paths) may not follow IGP SPT. Such paths may be chosen by a suitable network planning tool and provisioned on the source node of the SR-TE path.

It is possible to use a stateful PCE for computing one or more SR-TE paths taking into account various constraints and objective functions. Once a path is chosen, the stateful PCE can instantiate an SR-TE path on a PCC using PCEP extensions specified in [I-D.crabbe-pce-pce-initiated-lsp] using the SR specific PCEP extensions described in [I-D.sivabalan-pce-segment-routing].

By using the solutions provided from above drafts, LSP in both MPLS and GMPLS network can be setup/delete/maintained/synchronized through a centrally controlled dynamic MPLS network. Since in these solutions, the LSP is need to be signaled through the head end LER to the tail end LER, there are either RSVP-TE signaling protocol need to be deployed in the MPLS/GMPLS network, or extend TGP protocol with node/adjacency segment identifiers signaling capability to be deployed.

The PCECC solution proposed in this document allow for a dynamic MPLS network that is eventually controlled and deployed without the deployment of RSVP-TE protocol or extended IGP protocol with node/adjacency segment identifiers signaling capability while providing all the key MPLS functionalities needed by the service providers. These key MPLS features include MPLS P2P LSP, P2MP/MP2MP LSP, MPLS protection mechanism etc. In the case that one LSP path consists legacy network nodes and the new network nodes which are centrally controlled, the PCECC solution provides a smooth transition step for users.

1.2. Using the PCE as the Central Controller (PCECC) Approach

With PCECC, it not only removes the existing MPLS signaling totally from the control plane without losing any existing MPLS functionalities, but also PCECC achieves this goal through utilizing the existing PCEP without introducing a new protocol into the network.

The following diagram illustrates the PCECC architecture.

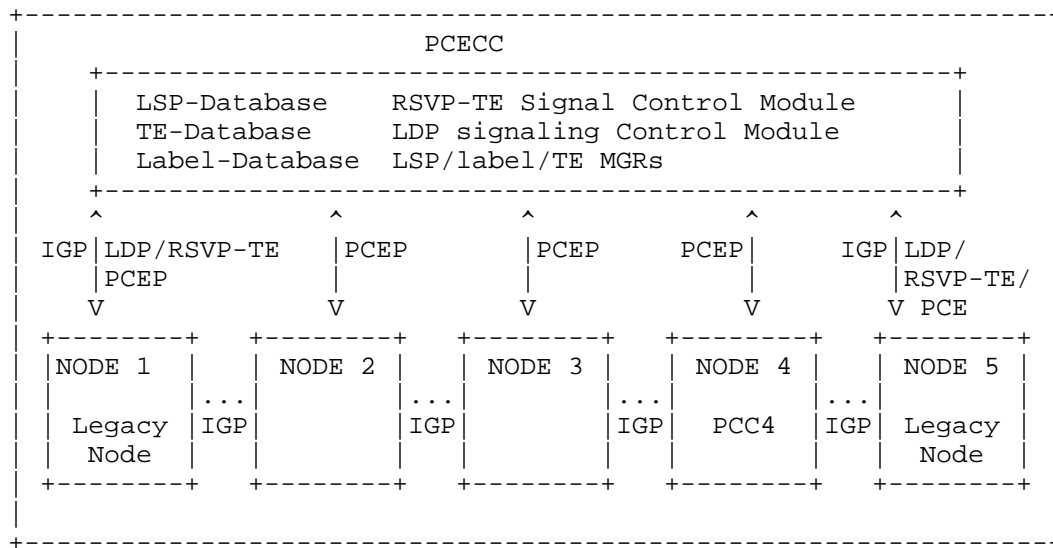


Figure 1: PCECC Architecture

Through the draft, we call the combination of the functionality for global label range signaling and the functionality of LSP setup/download/cleanup using the combination of global labels and local labels as PCECC functionality.

Current MPLS label has local meaning. That is, MPLS label allocated locally and signaled through the LDP/RSVP-TE/BGP etc dynamic signaling protocol.

As the SDN(Service-Driven Network) technology develops, MPLS global label has been proposed again for new solutions. [I-D.li-mpls-global-label-usecases] proposes possible usecases of MPLS global label. MPLS global label can be used for identification of the location, the service and the network in different application scenarios. From these usecases we can see that no matter SDN or traditional application scenarios, the new solutions based on MPLS global label can gain advantage over the existing solutions to facilitate service provisions. The solution choices are described in [I-D.li-mpls-global-label-framework].

To ease the label allocation and signaling mechanism, also with the new applications such as concentrated LSP controller is introduced, PCE can be conveniently used as a central controller and MPLS global label range negotiator.

The later section of this draft describes the user cases for PCE server and PCE clients to have the global label range negotiation and local label range negotiation functionality.

To empower networking with centralized controllable modules, there are many choices for downloading the forwarding entries to the data plane, one way is the use of the OpenFlow protocol, which helps devices populate their forwarding tables according to a set of instructions to the data plane. There are other candidate protocols to convey specific configuration information towards devices also. Since the PCEP protocol is already deployed in some of the service network, to leverage the PCEP to populated the MPLS forwarding table is a possible good choice.

For the centralized network, the performance achieved through distributed system can not be easy matched if all of the forwarding path is computed, downloaded and maintained by the centralized controller. The performance can be improved by supporting part of the forwarding path in the PCECC network through the segment routing mechanism except that the adjacency IDs for all the network nodes and links are propagated through the centralized controller instead of using the IGP extension.

The node and link adjacency IDs can be negotiated through the PCECC with each PCECC clients and these IDs can be just taken from the global label range which has been negotiated already.

With the capability of supporting SR within the PCECC architecture, all the p2p forwarding path protection use cases described in the draft [I-D.ietf-spring-resiliency-use-cases] will be supported too within the PCECC network. These protection alternatives include end-to-end path protection, local protection without operator management and local protection with operator management.

With the capability of global label and local label existing at the same time in the PCECC network, PCECC will use compute, setup and maintain the P2MP and MP2MP lsp using the local label range for each network nodes.

With the capability of setting up/maintaining the P2MP/MP2MP LSP within the PCECC network, it is easy to provide the end-end managed path protection service and the local protection with the operation management in the PCECC network for the P2MP/MP2MP LSP, which includes both the RSVP-TE P2MP based LSP and also the mLDP based LSP.

2. Terminology

The following terminology is used in this document.

IGP: Interior Gateway Protocol. Either of the two routing protocols, Open Shortest Path First (OSPF) or Intermediate System to Intermediate System (IS-IS).

PCC: Path Computation Client: any client application requesting a path computation to be performed by a Path Computation Element.

PCE: Path Computation Element. An entity (component, application, or network node) that is capable of computing a network path or route based on a network graph and applying computational constraints.

TE: Traffic Engineering.

3. PCEP Requirements

Following key requirements associated PCECC should be considered when designing the PCECC based solution:

1. Path Computation Element (PCE) clients supporting this draft MUST have the capability to advertise its PCECC capability to the PCECC.
2. Path Computation Element (PCE) supporting this draft MUST have the capability to negotiate a global label range for a group of clients.
3. Path Computation Client (PCC) MUST be able ask for global label range assigned in path request message .
4. PCE are not required to support label reserve service. Therefore, it MUST be possible for a PCE to reject a Path Computation Request message with a reason code that indicates no support for label reserve service.
5. PCEP SHOULD provide a means to return global label range and LSP label assignments of the computed path in the reply message.
6. PCEP SHOULD provide a means to download the MPLS forwarding entry to the PCECC's clients.

4. Use Cases of PCECC for Label Resource Reservations

Example 1 to 2 are based on network configurations illustrated using the following figure:

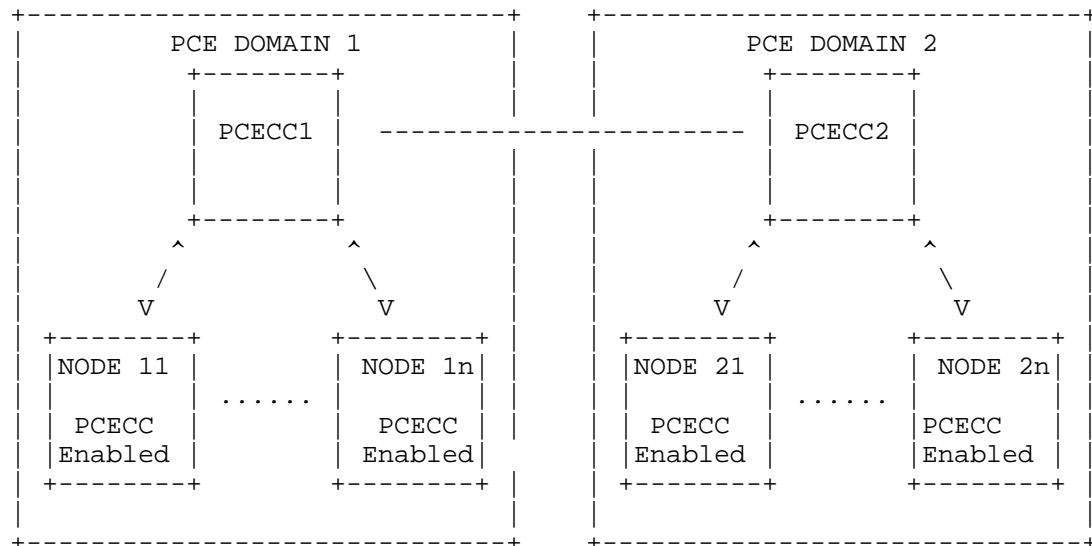


Figure 2: Using PCECC for Global Label Allocation

Example 1: Shared Global Label Range Reservation

- o PCECC Clients nodes report MPLS label capability to the central controller PCECC.
- o The central controller PCECC collects MPLS label capability of all nodes. Then PCECC can calculate the shared MPLS global label range for all the PCECC client nodes.
- o In the case that the shared global label range need to be negotiated across multiple domains, the central controllers of these domains need to be communicate to negotiate a common global label range.
- o The central controller PCECC notifies the shared global label range to all PCECC client nodes.

Example 2: Global Label Allocation

- o PCECC Client node1 send global label allocation request to the central controller PCECC1.
- o The central controller PCECC1 allocates the global label for FEC1 from the shared global label range and sends the reply to the client node1.
- o The central controller PCECC1 notifies the allocated label for FEC1 to all PCECC client nodes within domain 1.

5. Using PCECC for SR without the IGP Extension

For the centralized network, the performance achieved through distributed system can not be easily matched if all of the forwarding path is computed, downloaded and maintained by the centralized controller. The performance can be improved by supporting part of the forwarding path in the PCECC network through the segment routing mechanism except that node segment IDs and adjacency segment IDs for all the network are allocated dynamically and propagated through the centralized controller instead of using the IGP extension.

When the PCECC is used for the distribution of the node segment ID and adjacency segment ID, the node segment ID is allocated from the global label pool. For the allocation of adjacency segment ID, there are two choices, the first choice is that it is allocated from the local label pool, the second choice is that it is allocated from the global label pool. The advantage for the second choice is that the depth of the label stack for the forwarding path encoding will be reduced since adjacency segment ID can signal the forwarding path without adding the node segment ID in front of it. In this version of the draft, we use the first choice for now. We may update the draft to reflect the use of the second choice.

Same as the SR solutions, when PCECC is used as the central controller, the support of FRR on any topology can be pre-computed and setup without any additional signaling (other than the regular IGP/BGP protocols) including the support of shared risk constraints, support of node and link protection and support of microloop avoidance.

The following example illustrates the use case where the node segment ID and adjacency segment ID are allocated from the global label allocated for SR path.

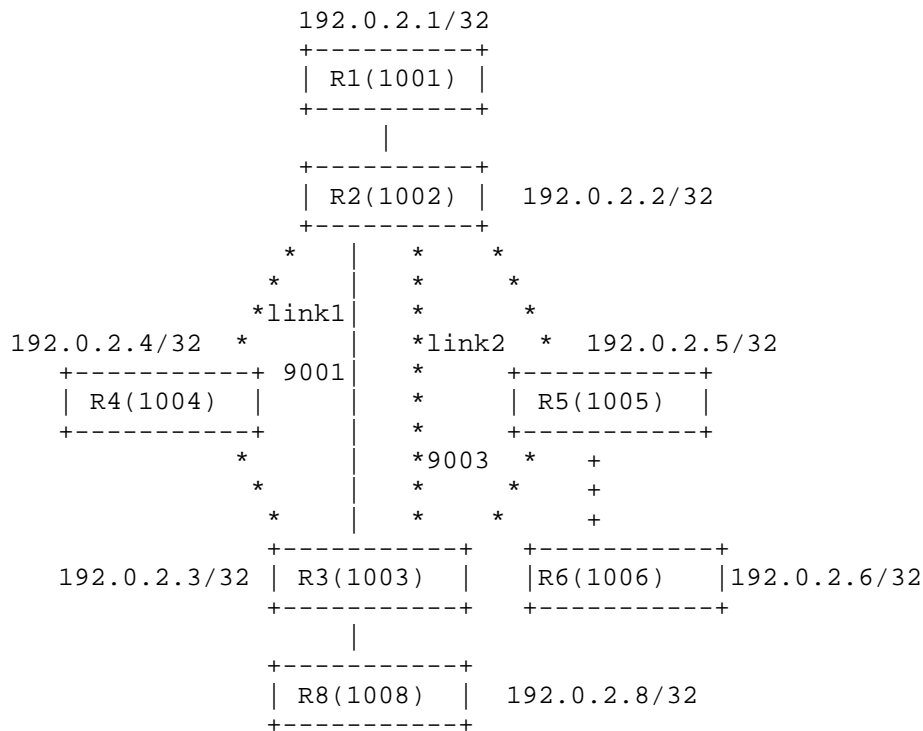


Figure 3: Using PCECC for SR Path

5.1. Use Cases of PCECC for SR Best Effort(BE) Path

In this mode of the solution, the PCECC just need to allocate the node segment ID and adjacency ID without calculating the explicit path for the SR path. The ingress of the forwarding path just need to encapsulate the destination node segment ID on top of the packet. All the intermediate nodes will forward the packet based on the final destination node segment id. It is similar to the LDP LSP forwarding except that label swapping is using the same global label both for the in segment and out segment in each hop.

The p2p SR BE path examples are explained as bellow:

Note that the node segment id for each node from the shared global labels ranges negotiated already.

Example 1:

R1 may send a packet to R8 simply by pushing an SR header with segment list {1008}. The path can be: R1-R2-R3-R8 or R1-R2-R5-R8

depending on the route calculation on node R2.

Example 2: local link/node protection:

For the packet which has destination of R3 and after that, R2 may preinstalled the backup forwarding entry to protect the R4 node, the pre-installed the backup path can go through either node5 or link1 or link2 between R2 and R3. The backup path calculation is locally decided by R2 and any existing IP FRR algorithms can be used here.

5.2. Use Cases of PCECC for SR Traffic Engineering (TE) Path

In the case of traffic engineering path is needed, the PCECC need to allocate the node segment ID and adjacency ID, and at the same time PCECC calculates the explicit path for the SR path and pass this explicit path represented with a sequence of node segment id and adjacency id. The ingress of the forwarding path need to encapsulate the stack of node segment id and adjacency id on top of the packet. For the case where strict traffic engineering path is needed, all the intermediate nodes and links will be specified through the stack of labels so that the packet is forwarded exactly as it is wanted.

Even though it is similar to TE LSP forwarding where forwarding path is engineered, but the Qos is only guaranteed through the enforce of the bandwidth admission control. As for the RSVP-TE LSP case, Qos is guaranteed through the link bandwidth reservation in each hop of the forwarding path.

The p2p SR traffic engineering path examples are explained as bellow:

Note that the node segment id for each node is allocated from the shared global labels ranges negotiated already and adjacency segment ids for each link are allocated from the local label pool for each node.

Example 1:

R1 may send a packet P1 to R8 simply by pushing an SR header with segment list {1008}. The path should be: R1-R2-R3-R8.

Example 2:

R1 may send a packet P2 to R8 by pushing an SR header with segment list {1002, 9001, 1008}. The path should be: R1-R2-(1)link-R3-R8.

Example 3:

R1 may send a packet P3 to R8 while avoiding the links between R2 and

R3 by pushing an SR header with segment list {1004, 1008}. The path should be : R1-R2-R4-R3-R8

The p2p local protection examples for SR TE path are explained as below:

Example 4: local link protection:

- o R1 may send a packet P4 to R8 by pushing an SR header with segment list {1002, 9001, 1008}. The path should be: R1-R2-(1)link-R3-R8.
- o When node R2 receives the packet from R1 which has the header of R2- (1)link-R3-R8, and also find out there is a link failure of link1, then it will send out the packet with header of R3-R8 through link2.

Example 5: local node protection:

- o R1 may send a packet P5 to R8 by pushing an SR header with segment list {1004, 1008}. The path should be : R1-R2-R4-R3-R8.
- o When node R2 receives the packet from R1 which has the header of {1004, 1008}, and also find out there is a node failure for node4, then it will send out the packet with header of {1005, 1008} to node5 instead of node4.

6. Use Cases of PCECC for TE LSP

In the previous sections, we have discussed the cases where the SR path is setup through the PCECC. Although those cases give the simplicity and scalability, but there are existing functionalities for the traffic engineering path such as the bandwidth guarantee through the full forwarding path and the multicast forwarding path which SR based solution cannot solve. Also there are cases where the depth of the label stack may have been an issue for existing deployment and certain vendors.

So to address these issues, PCECC architecture should also support the TE LSP and multicast LSP functionalities. To achieve this, the existing PCEP can be used to communicate between the PCE server and PCE's client PCC for exchanging the path request and reply information regarding to the TE LSP info. In this case, the TE LSP info is not only the path info itself, but it includes the full forwarding info. Instead of letting the ingress of LSP to initiate the LSP setup through the RSVP-TE signaling protocol, with minor extensions, we can use the PCEP to download the complete TE LSP forwarding entries for each node in the network.

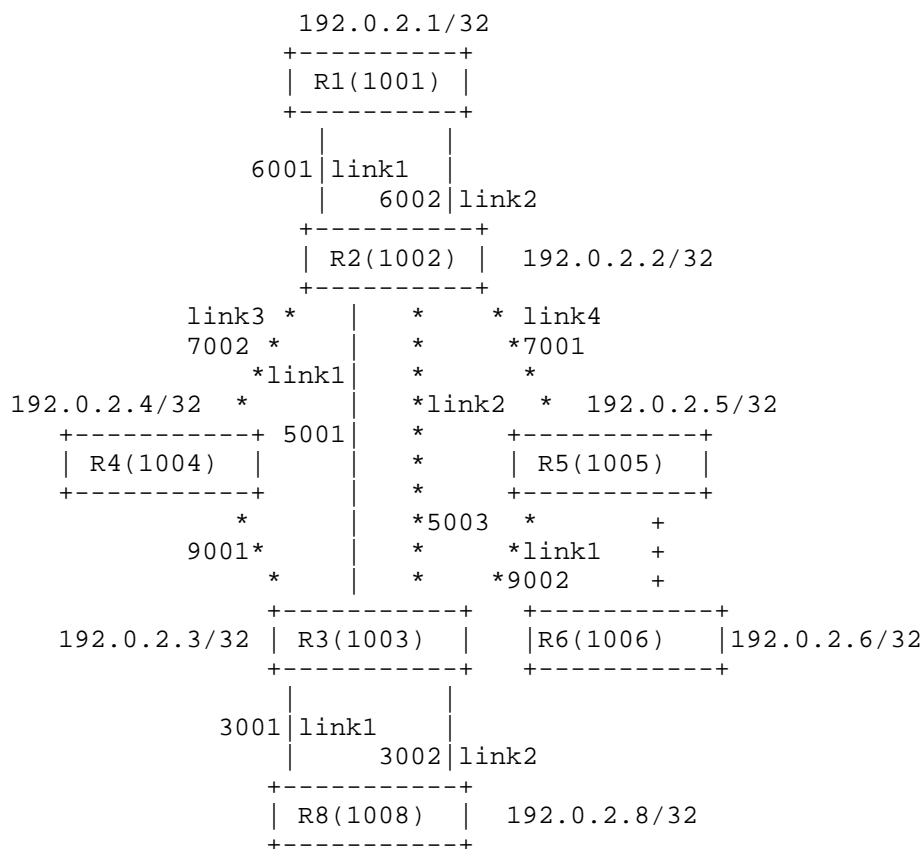


Figure 4: Using PCECC for TE LSP

TE LSP Setup Example

- o Node1 sends a path request message for the setup of TE LSP from R1 to R8.
- o PCECC program each node along the path from R1 to R8 with the primary path: {R1, link1, 6001}, {R2, link3, 7002}, {R4, link0, 9001}, {R3, link1, 3001}, {R8}.
- o For the end to end protection, PCECC program each node along the path from R1 to R8 with the secondary path: {R1, link2, 6002}, {R2, link4, 7001}, {R5, link1, 9002}, {R3, link2, 3002}, {R8}.
- o It is also possible to have a secondary backup path for the local node protection setup by PCECC. For example GBP[not] the primary path is still same as what we have setup so far, then to protect

the node R4 locally, PCECC can program the secondary path like this: {R1, link1, 6001}, {R2, link1, 5001}, {R3, link1, 3001}, {R8}. By doing this, the node R4 is locally protected.

7. Use Cases of PCECC for Multicast LSPs

The current multicast LSPs are setup either using the RSVP-TE P2MP or mLDP protocols. The setup of these LSPs not only need a lot of manual configurations, but also it is also complex when the protection is considered. By using the PCECC solution, the multicast LSP can be computed and setup through centralized controller which has the full picture of the topology and bandwidth usage for each link. It not only reduces the complex configurations comparing the distributed RSVP-TE P2MP or mLDP signal lings, but also it can compute the disjoint primary path and secondary path efficiently.

7.1. Using PCECC for P2MP/MP2MP LSPs' Setup

With the capability of global label and local label existing at the same time in the PCECC network, PCECC will use compute, setup and maintain the P2MP and MP2MP lsp using the local label range for each network nodes.

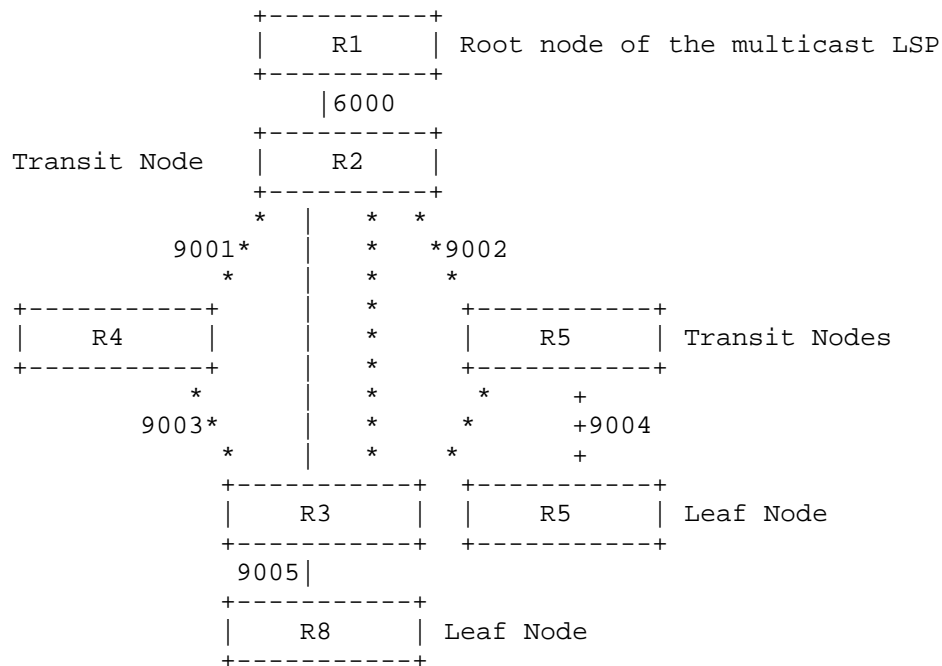


Figure 5: Using PCECC for P2MP TE LSP

The P2MP examples are explained here:

Step1: R1 may send a packet P1 to R2 simply by pushing an label of 6000 to the packet.

Step2: After R2 receives the packet with label 6000, it will forwarding to R4 by pushing header of 9001 and R5 by pushing header of 9002.

Step3: After R4 receives the packet with label 9001, it will forwarding to R3 by pushing header of 9003. After R5 receives the packet with label 9002, it will forwarding to R5 by pushing header of 9004.

Step3: After R3 receives the packet with label 9003, it will forwarding to R8 by pushing header of 9005

7.2. Use Cases of PCECC for the Resiliency of P2MP/MP2MP LSPs

7.2.1. PCECC for the End-to-End Protection of the P2MP/MP2MP LSPs

In this section we describe the end-end managed path protection service and the local protection with the operation management in the PCECC network for the P2MP/MP2MP LSP, which includes both the RSVP-TE P2MP based LSP and also the mLDP based LSP.

An end-to-end protection (for nodes and links) principle can be applied for computing backup P2MP or MP2MP LSPs. During computation of the primarily multicast trees, PCECC server may also be taken into consideration to compute a secondary tree. A PCE may compute the primary and backup P2MP or MP2MP LSP together or sequentially.

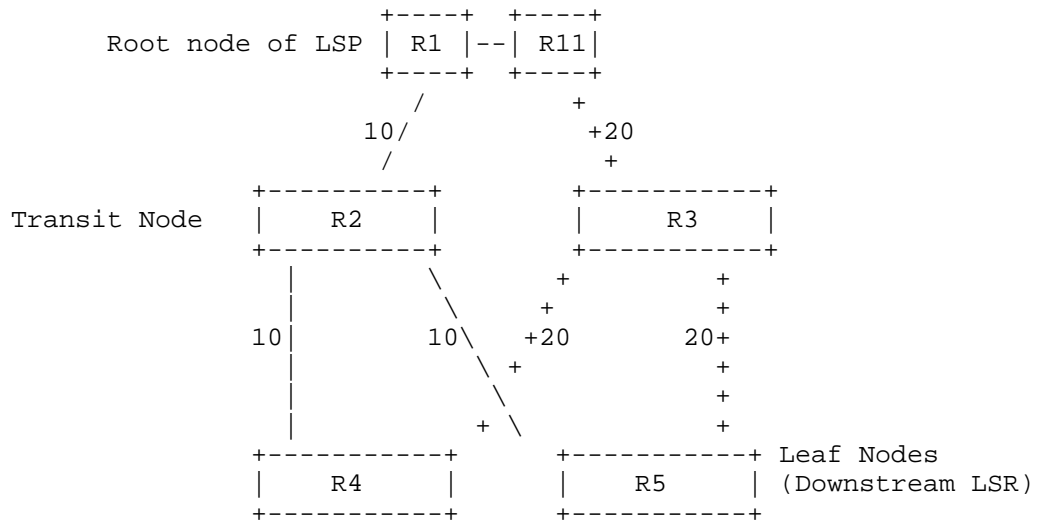


Figure 6: Using PCECC for P2MP TE End-to-End Protection

In the example above, when the PCECC setup the primary multicast tree from the root node R1 to the leafs, which is R1->R2->{R4, R5}, at same time, it can setup the backup tree, which is R11->R3->{R4, R5}. Both the these two primary forwarding tree and secondary forwarding tree will be downloaded to each routers along the primary path and the secondary path. The traffic will be forwarded through the R1->R2->{R4, R5} path normally, and when there is a node in the primary tree, then the root node R1 will switch the flow to the backup tree, which is R11->R3->{R4, R5}. By using the PCECC, the path computation and forwarding path downloading can all be done without the complex signaling used in the P2MP RSVP-TE or mLDP.

7.2.2. PCECC for the Local Protection of the P2MP/MP2MP LSPs

In this section we describe the local protection service in the PCECC network for the P2MP/MP2MP LSP.

While the PCECC sets up the primary multicast tree, it can also build the back LSP among PLR, the protected node, and MPs (the downstream nodes of the protected node). In the cases where the amount of downstream nodes are huge, this mechanism can avoid unnecessary packet duplication on PLR, so that protect the network from traffic congestion risk.

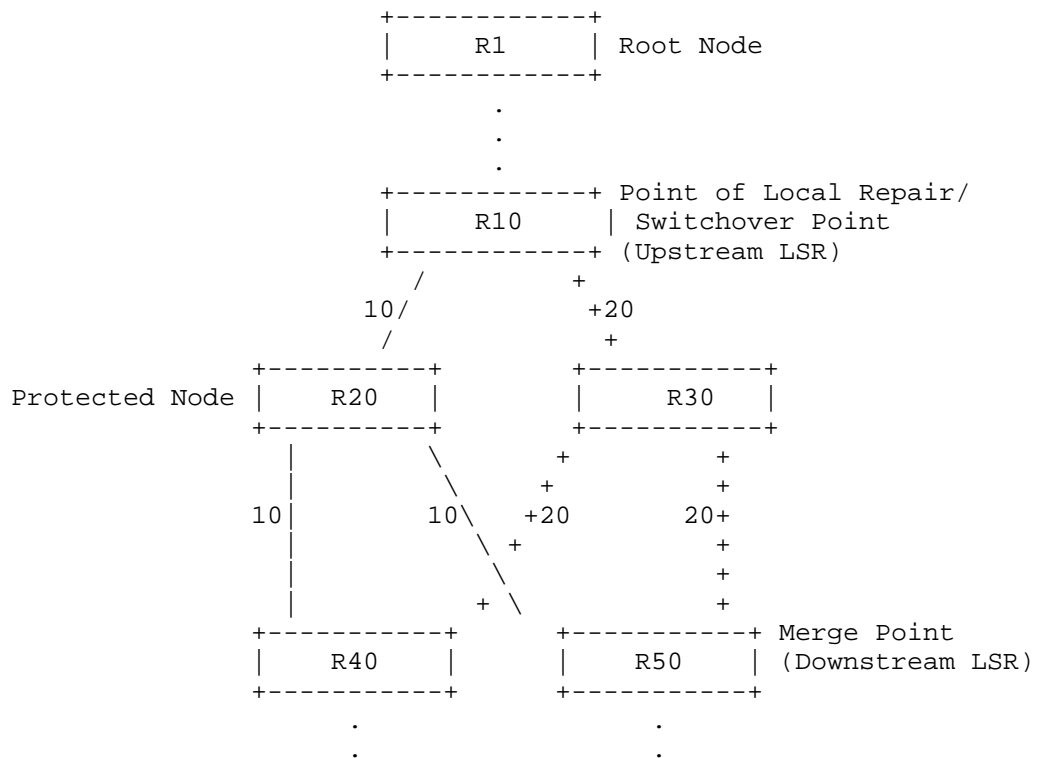


Figure 7: Using PCECC for P2MP TE Local Protection

In the example above, when the PCECC setup the primary multicast path around the PLR node R10 to protect node R20, which is R10->R20->{R40, R50}, at same time, it can setup the backup path R10->R30->{R40, R50}. Both the these two primary forwarding path and secondary forwarding path will be downloaded to each routers along the primary path and the secondary path. The traffic will be forwarded through the R10->R20->{R40, R50} path normally, and when there is a node failure for node R20, then the PLR node R10 will switch the flow to the backup path, which is R10->R30->{R40, R50}. By using the PCECC, the path computation and forwarding path downloading can all be done without the complex signaling used in the P2MP RSVP-TE or mLDLP.

8. Use Cases of PCECC for LSP in the Network Migration

One of the main advantages for PCECC solution is that it has backward compatibility naturally since the PCE server itself can function as a proxy node of MPLS network for all the new nodes which don't support the existing MPLS signaling protocol anymore.

As it is illustrated in the following example, the current network will migrate to a total PCECC controlled network gradually by replacing the legacy nodes. During the migration, the legacy nodes still need to signal using the existing MPLS protocol such as LDP and RSVP-TE, and the new nodes setup their portion of the forwarding path through PCECC directly. With the PCECC function as the proxy of these new nodes, MPLS signaling can populate through network as normal.

Example described in this section is based on network configurations illustrated using the following figure:

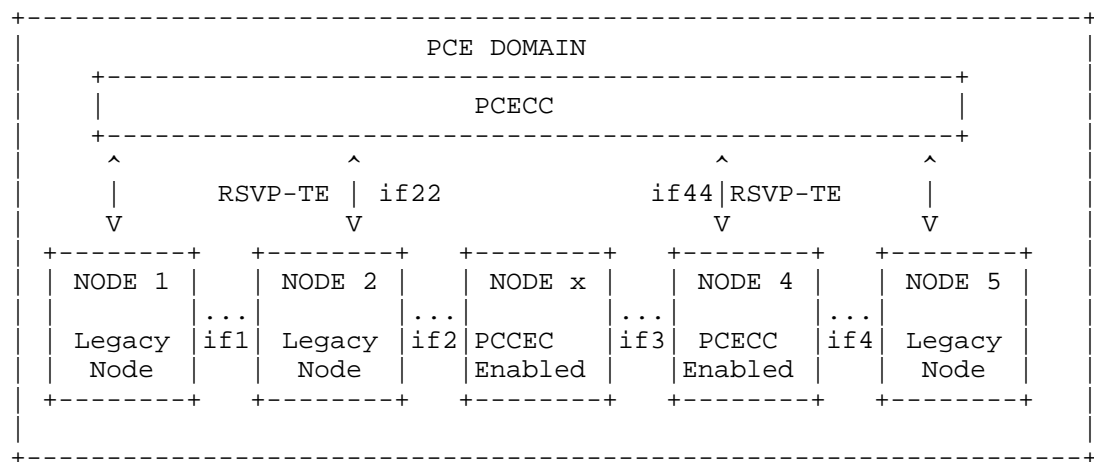


Figure 8: Using PCECC During Migration

Example: PCECC Initiated LSP Setup In the Network Migration

In this example, there are five nodes for the TE LSP from head end (node1) to the tail end (node5). Where the NodeX is central controlled and other nodes are legacy nodes.

- o Node1 sends a path request message for the setup of LSP destinating to Node5.
- o PCECC sends a reply message for LSP setup with path (node1, if1), (node2, if22), (node-PCECC, if44), (node4, if4), Nnode5.
- o Node1, Node2, Node-PCECC, Node 5 will setup the LSP to Node5 normally using the local label as normal.

- o Then the PCECC will program the outsegment of Node2, the insegment of Node4, and the insegment/outsegment for NodeX.

9. Use Cases of PCECC for L3VPN and PWE3

The existing services using MPLS LSP tunnels based on MPLS signalling mechanism such as L3VPN, PWE3 and IPv6 can be simplified by using the PCECC to negotiate the label assignments for the L3VPN, PWE3 and IPv6.

In the case of L3VPN, VPN labels can be negotiated and distributed through the PCECC PCEP among the PE router instead of using the BGP protocols.

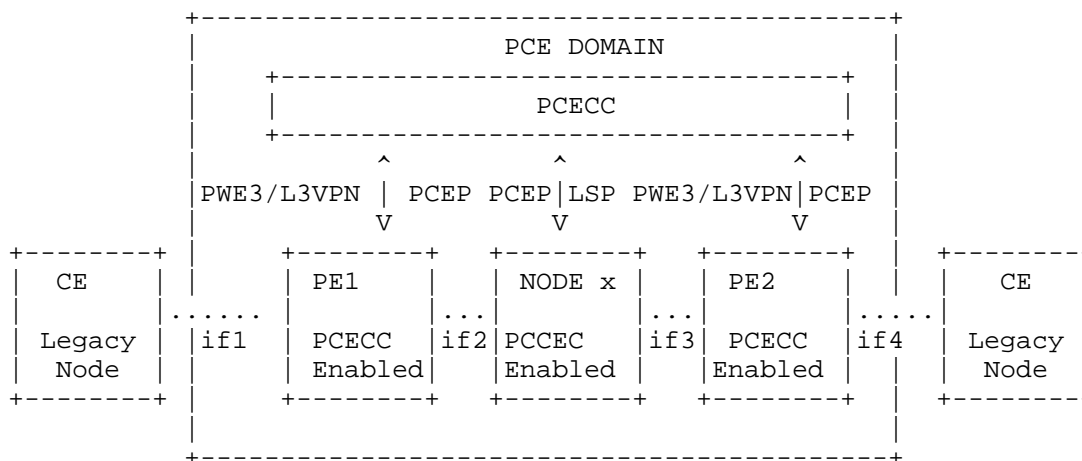


Figure 9: Using PCECC for L3VPN and PWE3

In the cast PWE3, instead of using the LDP signalling protocols, the lable and port pairs assigned to each pseudowire can be negotiated through PCECC among the PE rotuers and the corresponding forwarding entries will be distributed into each PE routers through the extended PCEP protocols.

10. The Considerations for PCECC Procedure and PCEP extensions

The PCECC's procedures and PCEP extensions is defined in [I-D.zhao-pce-pcep-extension-for-pce-controller].

11. IANA Considerations

This document does not require any action from IANA.

12. Security Considerations

TBD.

13. Acknowledgments

We would like to thank Robert Tao, Changjiang Yan, Tieying Huang for their useful comments and suggestions.

14. References

14.1. Normative References

- | | |
|-----------|---|
| [RFC2119] | Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997. |
| [RFC5440] | Vasseur, JP. and JL. Le Roux, "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, March 2009. |

14.2. Informative References

- | | |
|-----------|---|
| [RFC5441] | Vasseur, JP., Zhang, R., Bitar, N., and JL. Le Roux, "A Backward-Recursive PCE-Based Computation (BRPC) Procedure to Compute Shortest Constrained Inter-Domain Traffic Engineering Label Switched Paths", RFC 5441, |
|-----------|---|

April 2009.

[RFC5541]

Le Roux, J.L.,
Vasseur, J.P., and
Y. Lee, "Encoding
of Objective
Functions in the
Path Computation
Element
Communication
Protocol (PCEP)",
RFC 5541,
June 2009.

[I-D.filsfils-spring-segment-routing]

Filsfils, C.,
Previdi, S.,
Bashandy, A.,
Decraene, B.,
Litkowski, S.,
Horneffer, M.,
Milojevic, I.,
Shakir, R., Ytti,
S., Henderickx, W.,
Tantsura, J., and
E. Crabbe, "Segment
Routing
Architecture", draf
t-filsfils-spring-
segment-routing-04
(work in progress),
July 2014.

[I-D.ietf-pce-stateful-pce]

Crabbe, E., Minei,
I., Medved, J., and
R. Varga, "PCEP
Extensions for
Stateful PCE", draf
t-ietf-pce-
stateful-pce-11
(work in progress),
April 2015.

[I-D.crabbe-pce-pce-initiated-lsp]

Crabbe, E., Minei,
I., Sivabalan, S.,
and R. Varga, "PCEP
Extensions for PCE-
initiated LSP Setup
in a Stateful PCE

Model", draft-crabbe-pce-pce-initiated-lsp-03 (work in progress), October 2013.

[I-D.ali-pce-remote-initiated-gmpls-lsp]

Ali, Z., Sivabalan, S., Filsfils, C., Varga, R., Lopez, V., Dios, O., and X. Zhang, "Path Computation Element Communication Protocol (PCEP) Extensions for remote-initiated GMPLS LSP Setup", draft-ali-pce-remote-initiated-gmpls-lsp-03 (work in progress), February 2014.

[I-D.ietf-isis-segment-routing-extensions]

Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-05 (work in progress), June 2015.

[I-D.psenak-ospf-segment-routing-extensions]

Psenak, P., Previdi, S., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", draft-psenak-ospf-

- segment-routing-extensions-05 (work in progress), June 2014.
- [I-D.sivabalan-pce-segment-routing] Sivabalan, S., Medved, J., Filsfils, C., Crabbe, E., Raszuk, R., Lopez, V., and J. Tantsura, "PCEP Extensions for Segment Routing", draft-sivabalan-pce-segment-routing-03 (work in progress), July 2014.
- [I-D.li-mpls-global-label-usecases] Li, Z., Zhao, Q., Yang, T., and R. Raszuk, "Use Cases of MPLS Global Label", draft-li-mpls-global-label-usecases-02 (work in progress), July 2014.
- [I-D.li-mpls-global-label-framework] Li, Z., Zhao, Q., Chen, X., Yang, T., and R. Raszuk, "A Framework of MPLS Global Label", draft-li-mpls-global-label-framework-02 (work in progress), July 2014.
- [I-D.zhao-pce-pcep-extension-for-pce-controller] Zhao, Q., Zhao, K., Li, Z., Dhody, D., Palle, U., and T. Communications, "PCEP Procedures and Protocol Extensions for Using PCE as a Central Controller (PCECC) of LSPs", d

raft-zhao-pce-pcep-
extension-for-pce-
controller-01 (work
in progress),
March 2015.

[I-D.ietf-spring-resiliency-use-cases]

Francois, P.,
Filsfils, C.,
Decraene, B., and
R. Shakir, "Use-
cases for
Resiliency in
SPRING", draft-
ietf-spring-
resiliency-use-
cases-01 (work in
progress),
March 2015.

Authors' Addresses

Quintin Zhao
Huawei Technologies
125 Nagog Technology Park
Acton, MA 01719
US

EMail: quintin.zhao@huawei.com

Robin Li
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

EMail: lizhenbin@huawei.com

King Ke
Tencent Holdings Ltd.
Shenzhen
China

EMail: kinghe@tencent.com

Luyuan Fang
Microsoft

EMail: lufang@microsoft.com

Chao Zhou
Cisco Systems

EMail: chao.zhou@cisco.com

Boris Zhang
Telus Communications
200 Consilium Pl Floor 15
Toronto, ON M1H 3J3
Canada

EMail: Boris.Zhang@telus.com

Katherine Zhao
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

EMail: Katherine.zhao@huawei.com

Richard Li
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

EMail: renwei.li@huawei.com

PCE Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 20, 2018

Q. Zhao
Z. Li
D. Dhody
S. Karunanithi
Huawei Technologies
A. Farrel
Juniper Networks, Inc
C. Zhou
Cisco Systems
June 18, 2018

PCEP Procedures and Protocol Extensions for Using PCE as a Central
Controller (PCECC) of LSPs
draft-zhao-pce-pcep-extension-for-pce-controller-08

Abstract

The Path Computation Element (PCE) is a core component of Software-Defined Networking (SDN) systems. It can compute optimal paths for traffic across a network and can also update the paths to reflect changes in the network or traffic demands.

PCE was developed to derive paths for MPLS Label Switched Paths (LSPs), which are supplied to the head end of the LSP using the Path Computation Element Communication Protocol (PCEP). But SDN has a broader applicability than signaled (G)MPLS traffic-engineered (TE) networks, and the PCE may be used to determine paths in a range of use cases. PCEP has been proposed as a control protocol for use in these environments to allow the PCE to be fully enabled as a central controller.

A PCE-based central controller (PCECC) can simplify the processing of a distributed control plane by blending it with elements of SDN and without necessarily completely replacing it. Thus, the LSP can be calculated/setup/initiated and the label forwarding entries can also be downloaded through a centralized PCE server to each network devices along the path while leveraging the existing PCE technologies as much as possible.

This document specifies the procedures and PCEP protocol extensions for using the PCE as the central controller.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 20, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
1.1. Requirements Language	5
2. Terminology	5
3. Basic PCECC Mode	5
4. PCEP Requirements	5
5. Procedures for Using the PCE as the Central Controller (PCECC)	6
5.1. Stateful PCE Model	6
5.2. New LSP Functions	6
5.3. PCECC Capability Advertisement	7
5.4. LSP Operations	8
5.4.1. Basic PCECC LSP Setup	8
5.4.2. Central Control Instructions	10
5.4.2.1. Label Download	10
5.4.2.2. Label Cleanup	11
5.4.3. PCE Initiated PCECC LSP	12
5.4.4. PCECC LSP Update	14
5.4.5. Re Delegation and Cleanup	16
5.4.6. Synchronization of Central Controllers Instructions .	16

5.4.7. PCECC LSP State Report	16
6. PCEP messages	16
6.1. The PCInitiate message	17
6.2. The PCRpt message	18
7. PCEP Objects	19
7.1. OPEN Object	19
7.1.1. PCECC Capability sub-TLV	19
7.2. PATH-SETUP-TYPE TLV	20
7.3. CCI Object	20
7.3.1. Address TLVs	21
8. Security Considerations	23
8.1. Malicious PCE	23
9. Manageability Considerations	23
9.1. Control of Function and Policy	23
9.2. Information and Data Models	23
9.3. Liveness Detection and Monitoring	23
9.4. Verify Correct Operations	23
9.5. Requirements On Other Protocols	23
9.6. Impact On Network Operations	24
10. IANA Considerations	24
10.1. PCEP TLV Type Indicators	24
10.2. New Path Setup Type Registry	24
10.3. PCEP Object	24
10.4. CCI Object Flag Field	24
10.5. PCEP-Error Object	25
11. Acknowledgments	25
12. References	25
12.1. Normative References	25
12.2. Informative References	26
Appendix A. Contributor Addresses	29
Authors' Addresses	29

1. Introduction

The Path Computation Element (PCE) [RFC4655] was developed to offload path computation function from routers in an MPLS traffic-engineered network. Since then, the role and function of the PCE has grown to cover a number of other uses (such as GMPLS [RFC7025]) and to allow delegated control [RFC8231] and PCE-initiated use of network resources [RFC8281].

According to [RFC7399], Software-Defined Networking (SDN) refers to a separation between the control elements and the forwarding components so that software running in a centralized system, called a controller, can act to program the devices in the network to behave in specific ways. A required element in an SDN architecture is a component that plans how the network resources will be used and how the devices will be programmed. It is possible to view this

component as performing specific computations to place traffic flows within the network given knowledge of the availability of network resources, how other forwarding devices are programmed, and the way that other flows are routed. This is the function and purpose of a PCE, and the way that a PCE integrates into a wider network control system (including an SDN system) is presented in [RFC7491].

In early PCE implementations, where the PCE was used to derive paths for MPLS Label Switched Paths (LSPs), paths were requested by network elements (known as Path Computation Clients (PCCs)), and the results of the path computations were supplied to network elements using the Path Computation Element Communication Protocol (PCEP) [RFC5440]. This protocol was later extended to allow a PCE to send unsolicited requests to the network for LSP establishment [RFC8281].

[RFC8283] introduces the architecture for PCE as a central controller as an extension of the architecture described in [RFC4655] and assumes the continued use of PCEP as the protocol used between PCE and PCC. [RFC8283] further examines the motivations and applicability for PCEP as a Southbound Interface (SBI), and introduces the implications for the protocol. [I-D.ietf-teas-pcecc-use-cases] describes the use cases for the PCECC architecture.

A PCE-based central controller (PCECC) can simplify the processing of a distributed control plane by blending it with elements of SDN and without necessarily completely replacing it. Thus, the LSP can be calculated/setup/initiated and the label forwarding entries can also be downloaded through a centralized PCE server to each network devices along the path while leveraging the existing PCE technologies as much as possible.

This draft specify the procedures and PCEP protocol extensions for using the PCE as the central controller for static LSPs, where LSPs can be provisioned as explicit label instructions at each hop on the end-to-end path. Each router along the path must be told what label-forwarding instructions to program and what resources to reserve. The PCE-based controller keeps a view of the network and determines the paths of the end-to-end LSPs, and the controller uses PCEP to communicate with each router along the path of the end-to-end LSP.

The extension for PCECC in Segment Routing (SR) is specified in a separate draft [I-D.zhao-pce-pcep-extension-pce-controller-sr].

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Terminology

Terminologies used in this document is same as described in the draft [RFC8283] and [I-D.ietf-teas-pcecc-use-cases].

3. Basic PCECC Mode

In this mode LSPs are provisioned as explicit label instructions at each hop on the end-to-end path. Each router along the path must be told what label forwarding instructions to program and what resources to reserve. The controller uses PCEP to communicate with each router along the path of the end-to-end LSP.

Note that the PCE-based controller will take responsibility for managing some part of the MPLS label space for each of the routers that it controls, and may take wider responsibility for partitioning the label space for each router and allocating different parts for different uses. This is also described in section 3.1.2. of [RFC8283]. For the purpose of this document, it is assumed that label range to be used by a PCE is known and set on both PCEP peers. A future extension could add this capability to advertise the range via possible PCEP extensions as well. The rest of processing is similar to the existing stateful PCE mechanism.

4. PCEP Requirements

Following key requirements associated PCECC should be considered when designing the PCECC based solution:

1. PCEP speaker supporting this draft MUST have the capability to advertise its PCECC capability to its peers.
2. PCEP speaker not supporting this draft MUST be able to reject PCECC related extensions with a error reason code that indicates that this feature is not supported.
3. PCEP speaker MUST provide a means to identify PCECC based LSP in the PCEP messages.

4. PCEP procedures SHOULD provide a means to update (or cleanup) the label- download entry to the PCC.
 5. PCEP procedures SHOULD provide a means to synchronize the labels between PCE to PCC in PCEP messages.
5. Procedures for Using the PCE as the Central Controller (PCECC)
- 5.1. Stateful PCE Model

Active stateful PCE is described in [RFC8231]. PCE as a central controller (PCECC) reuses existing Active stateful PCE mechanism as much as possible to control the LSP.

5.2. New LSP Functions

This document defines the following new PCEP messages and extends the existing messages to support PCECC:

(PCRpt): a PCEP message described in [RFC8231]. PCRpt message is used to send PCECC LSP Reports. It is also extended to report the set of Central Controller's Instructions (CCI) (label forwarding instructions in the context of this document) received from the PCE. See Section 5.4.6 for more details.

(PCInitiate): a PCEP message described in [RFC8281]. PCInitiate message is used to setup PCE-Initiated LSP based on PCECC mechanism. It is also extended for Central Controller's Instructions (CCI) (download or cleanup the Label forwarding instructions in the context of this document) on all nodes along the path.

(PCUpd): a PCEP message described in [RFC8231]. PCUpd message is used to send PCECC LSP Update.

The new LSP functions defined in this document are mapped onto the messages as shown in the following table.

Function	Message
PCECC Capability advertisement	Open
Label entry Add	PCInitiate
Label entry Cleanup	PCInitiate
PCECC Initiated LSP	PCInitiate
PCECC LSP Update	PCUpd
PCECC LSP State Report	PCRpt
PCECC LSP Delegation	PCRpt
PCECC Label Report	PCRpt

This document specify a new object CCI (see Section 7.3) for the encoding of central controller's instructions. In the scope of this document this is limited to Label forwarding instructions. The CC-ID is the unique identifier for the central controller's instructions in PCEP. The PCEP messages are extended in this document to handle the PCECC operations.

5.3. PCECC Capability Advertisement

During PCEP Initialization Phase, PCEP Speakers (PCE or PCC) advertise their support of PCECC extensions.

This document defines a new Path Setup Type (PST) [I-D.ietf-pce-lsp-setup-type] for PCECC, as follows:

- o PST = TBD: Path is setup via PCECC mode.

A PCEP speaker MUST indicate its support of the function described in this document by sending a PATH-SETUP-TYPE-CAPABILITY TLV in the OPEN object with this new PST included in the PST list.

This document also defines the PCECC Capability sub-TLV Section 7.1.1. PCEP speakers use this sub-TLV to exchange information about their PCECC capability. If a PCEP speaker includes PST=TBD in the PST List of the PATH-SETUP-TYPE-CAPABILITY TLV then it MUST also include the PCECC Capability sub-TLV inside the PATH-SETUP-TYPE-CAPABILITY TLV.

The presence of the PST and PCECC Capability sub-TLV in PCC's OPEN Object indicates that the PCC is willing to function as a PCECC client.

The presence of the PST and PCECC Capability sub-TLV in PCE's OPEN message indicates that the PCE is interested in function as a PCECC server.

The PCEP protocol extensions for PCECC MUST NOT be used if one or both PCEP Speakers have not included the PST or the PCECC Capability sub-TLV in their respective OPEN message. If the PCEP Speakers support the extensions of this draft but did not advertise this capability then a PCerr message with Error-Type=19(Invalid Operation) and Error-Value=TBD (Attempted PCECC operations when PCECC capability was not advertised) will be generated and the PCEP session will be terminated.

A PCC or a PCE MUST include both PCECC-CAPABILITY sub-TLV and STATEFUL-PCE-CAPABILITY TLV ([RFC8231]) (with I flag set [RFC8281]) in OPEN Object to support the extensions defined in this document. If PCECC-CAPABILITY sub-TLV is advertised and STATEFUL-PCE-CAPABILITY TLV is not advertised in OPEN Object, it SHOULD send a PCerr message with Error-Type=19 (Invalid Operation) and Error-value=TBD (stateful PCE capability was not advertised) and terminate the session.

5.4. LSP Operations

The PCEP messages pertaining to PCECC MUST include PATH-SETUP-TYPE TLV [I-D.ietf-pce-lsp-setup-type] in the SRP object to clearly identify the PCECC LSP is intended.

5.4.1. Basic PCECC LSP Setup

In order to setup a LSP based on PCECC mechanism, a PCC MUST delegate the LSP by sending a PCRpt message with PST set for PCECC (see Section 7.2) and D (Delegate) flag (see [RFC8231]) set in the LSP object.

LSP-IDENTIFIER TLV MUST be included for PCECC LSP, the tuple uniquely identifies the LSP in the network. The LSP object is included in central controller's instructions (label download) to identify the PCECC LSP for this instruction. The PLSP-ID is the original identifier used by the ingress PCC, so the transit LSR could have multiple central controller instructions that have the same PLSP-ID. The PLSP-ID in combination with the source (in LSP-IDENTIFIER TLV) MUST be unique. The PLSP-ID is included for maintainability reasons. As per [RFC8281], the LSP object could include SPEAKER-ENTITY-ID TLV to identify the PCE that initiated these instructions. Also the CC-ID is unique on the PCEP session as described in Section 7.3.

When a PCE receives PCRpt message with D flags and PST Type set, it calculates the path and assigns labels along the path; and set up the

path by sending PCInitiate message to each node along the path of the LSP. The PCC generates a Path Computation State Report (PCRpt) and include the central controller's instruction (CCI) and the identified LSP. The CC-ID is uniquely identify the central controller's instruction within PCEP. The PCC further responds with the PCRpt messages including the CCI and LSP objects.

Once the central controller's instructions (label operations) are completed, the PCE SHOULD send the PCUpd message to the Ingress PCC. The PCUpd message is as per [RFC8231] SHOULD include the path information as calculated by the PCE.

Note that the PCECC LSPs MUST be delegated to a PCE at all times.

LSP deletion operation for PCECC LSP is same as defined in [RFC8231]. If the PCE receives PCRpt message for LSP deletion then it does Label cleanup operation as described in Section 5.4.2.2 for the corresponding LSP.

The Basic PCECC LSP setup sequence is as shown below.

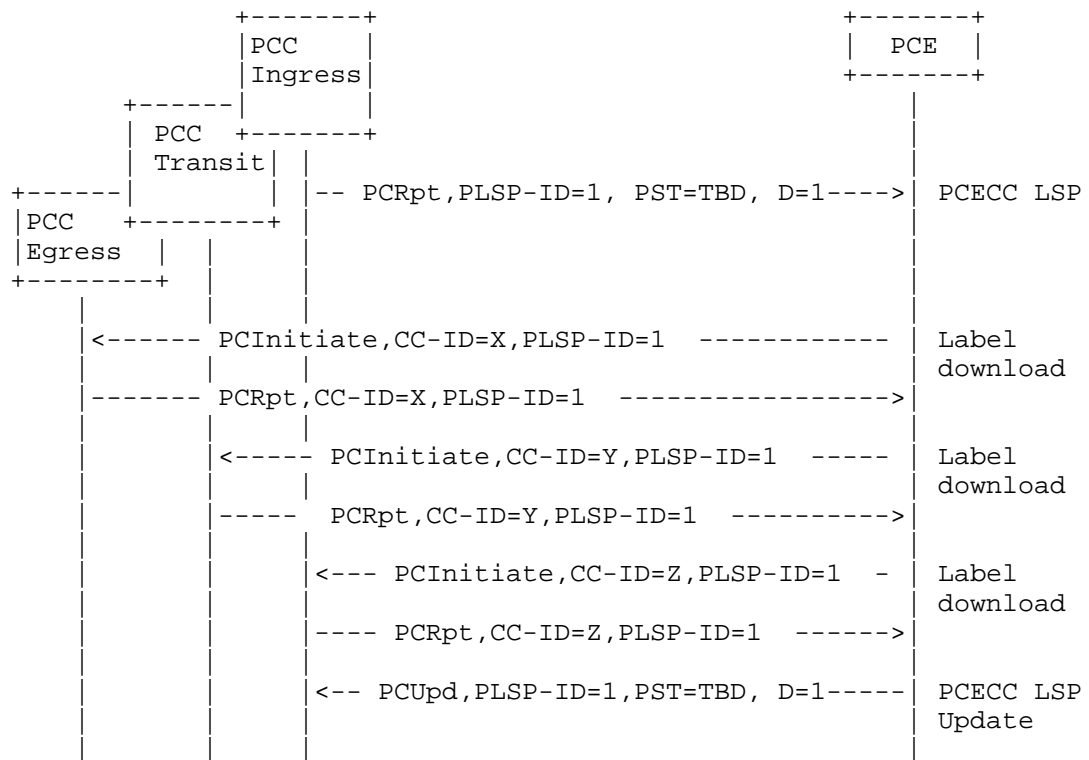


Figure 2: Basic PCECC LSP setup

The PCECC LSP are considered to be 'up' by default (on receipt of PCUpd message from PCE). The Ingress MAY further choose to deploy a data plane check mechanism and report the status back to the PCE via PCRpt message.

5.4.2. Central Control Instructions

The new central controller's instructions (CCI) for the label operations in PCEP is done via the PCInitiate message, by defining a new PCEP Objects for CCI operations. Local label range of each PCC is assumed to be known at both the PCC and the PCE.

5.4.2.1. Label Download

In order to setup an LSP based on PCECC, the PCE sends a PCInitiate message to each node along the path to download the Label instruction as described in Section 5.4.1.

The CCI object MUST be included, along with the LSP object in the PCInitiate message. The LSP-IDENTIFIER TLV MUST be included in LSP object. The SPEAKER-ENTITY-ID TLV SHOULD be included in LSP object.

If a node (PCC) receives a PCInitiate message which includes a Label to download as part of CCI, that is out of the range set aside for the PCE, it MUST send a PCErr message with Error-type=TBD (PCECC failure) and Error-value=TBD (Label out of range) and MUST include the SRP object to specify the error is for the corresponding label update via PCInitiate message. If a PCC receives a PCInitiate message but failed to download the Label entry, it MUST send a PCErr message with Error-type=TBD (PCECC failure) and Error-value=TBD (instruction failed) and MUST include the SRP object to specify the error is for the corresponding label update via PCInitiate message.

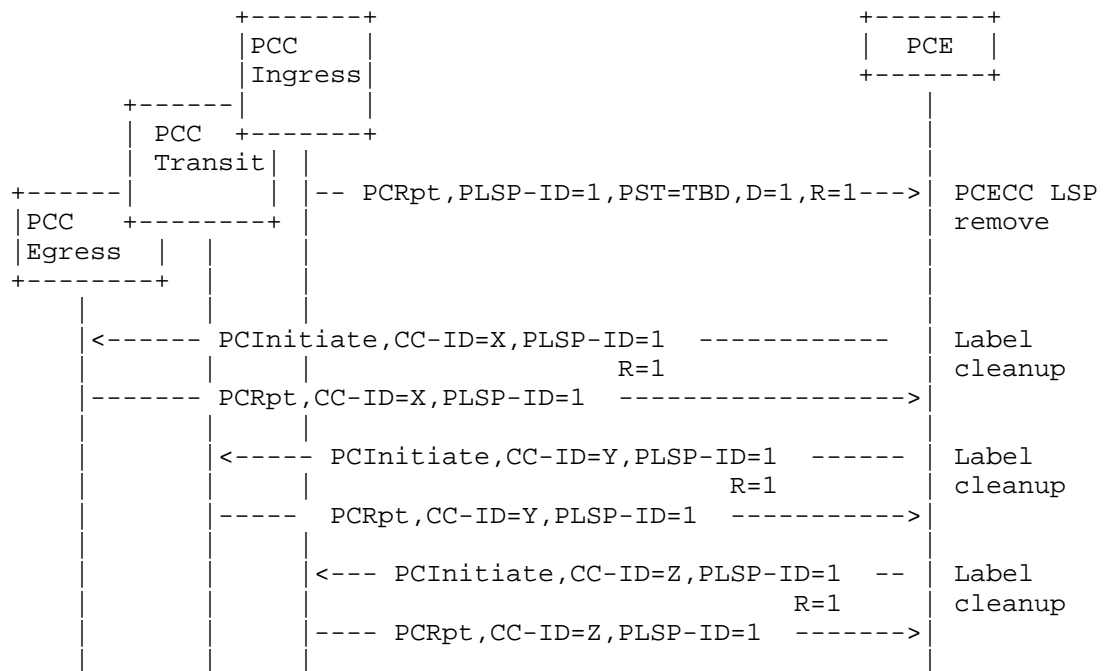
New PCEP object for central control instructions (CCI) is defined in Section 7.3.

5.4.2.2. Label Cleanup

In order to delete an LSP based on PCECC, the PCE sends a central controller instructions via a PCInitiate message to each node along the path of the LSP to cleanup the Label forwarding instruction.

If the PCC receives a PCInitiate message but does not recognize the label in the CCI, the PCC MUST generate a PCErr message with Error-Type 19(Invalid operation) and Error-Value=TBD, "Unknown Label" and MUST include the SRP object to specify the error is for the corresponding label cleanup (via PCInitiate message).

The R flag in the SRP object defined in [RFC8281] specifies the deletion of Label Entry in the PCInitiate message.



As per [RFC8281], following the removal of the Label forwarding instruction, the PCC MUST send a PCRpt message. The SRP object in the PCRpt MUST include the SRP-ID-number from the PCInitiate message that triggered the removal. The R flag in the SRP object MUST be set.

5.4.3. PCE Initiated PCECC LSP

The LSP Instantiation operation is same as defined in [RFC8281].

In order to setup a PCE Initiated LSP based on the PCECC mechanism, a PCE sends PCInitiate message with Path Setup Type set for PCECC (see Section 7.2) to the Ingress PCC.

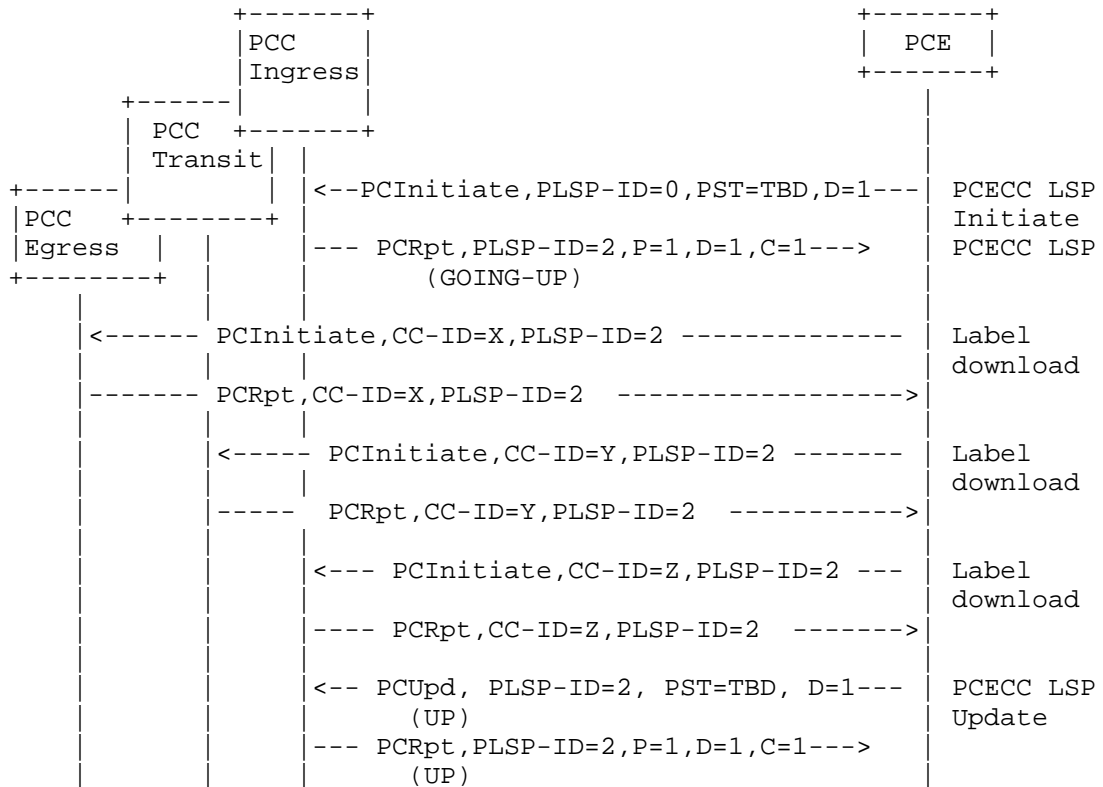
The Ingress PCC MUST also set D (Delegate) flag (see [RFC8231]) and C (Create) flag (see [RFC8281]) in LSP object of PCRpt message. The PCC responds with first PCRpt message with the status as "GOING-UP" and assigned PLSP-ID.

Note that the label forwarding instructions from PCECC are send after the initial PCInitiate and PCRpt exchange. This is done so that the PLSP-ID and other LSP identifiers can be obtained from the ingress and can be included in the label forwarding instruction in the next

PCInitiate message. The rest of the PCECC LSP setup operations are same as those described in Section 5.4.1.

The LSP deletion operation for PCE Initiated PCECC LSP is same as defined in [RFC8281]. The PCE should further perform Label entry cleanup operation as described in Section 5.4.2.2 for the corresponding LSP.

The PCE Initiated PCECC LSP setup sequence is shown below -

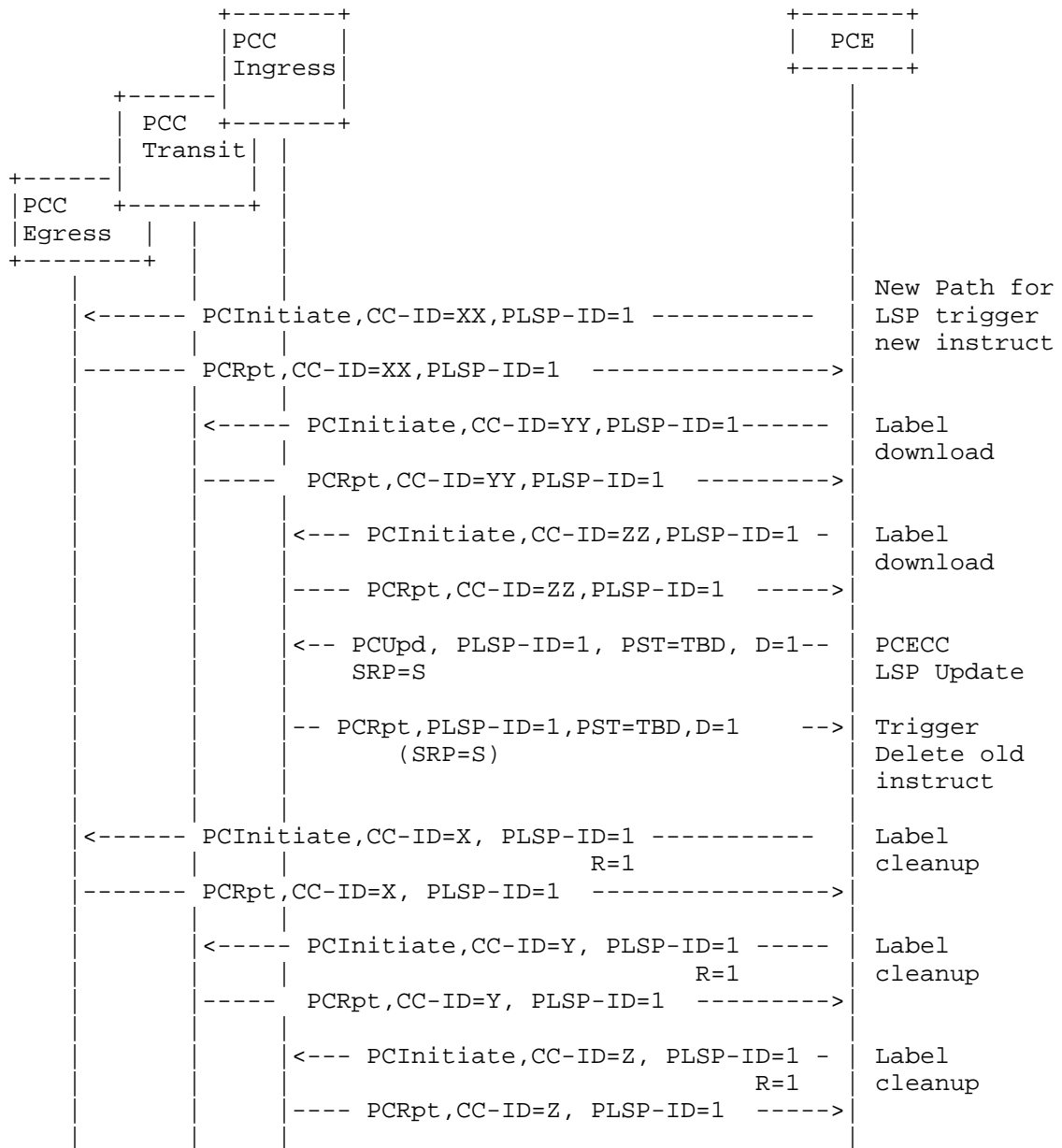


Once the label operations are completed, the PCE SHOULD send the PCUpd message to the Ingress PCC. The PCUpd message is as per [RFC8231].

5.4.4. PCECC LSP Update

In case of a modification of PCECC LSP with a new path, a PCE sends a PCUpd message to the Ingress PCC. But to follow the make-before-break procedures, the PCECC first update new instructions based on the updated LSP and then update to ingress to switch traffic, before cleaning up the old instructions. A new CC-ID is used to identify the updated instruction, the existing identifiers in the LSP object identify the existing LSP. Once new instructions are downloaded, the PCE further updates the new path at the ingress which triggers the traffic switch on the updated path. The Ingress PCC acknowledges with a PCRpt message, on receipt of PCRpt message, the PCE does cleanup operation for the old LSP as described in Section 5.4.2.2.

The PCECC LSP Update sequence is shown below -



The modified PCECC LSP are considered to be 'up' by default. The Ingress MAY further choose to deploy a data plane check mechanism and report the status back to the PCE via PCRpt message.

5.4.5. Re Delegation and Cleanup

As described in [RFC8281], a new PCE can gain control over the orphaned LSP. In case of PCECC LSP, the new PCE MUST also gain control over the central controllers instructions in the same way by sending a PCInitiate message that includes the SRP, LSP and CCI objects and carries the CC-ID and PLSP-ID identifying the instruction, it wants to take control of.

Further, as described in [RFC8281], the State Timeout Interval timer ensures that a PCE crash does not result in automatic and immediate disruption for the services using PCE-initiated LSPs. Similarly the central controller instructions are not removed immediately upon PCE failure. Instead, they are cleaned up on the expiration of this timer. This allows for network cleanup without manual intervention. The PCC MUST support removal of CCI as one of the behaviors applied on expiration of the State Timeout Interval timer.

5.4.6. Synchronization of Central Controllers Instructions

The purpose of Central Controllers Instructions synchronization (labels in the context of this document) is to make sure that the PCE's view of CCI (Labels) matches with the PCC's Label allocation. This synchronization is performed as part of the LSP state synchronization as described in [RFC8231] and [RFC8233].

As per LSP State Synchronization [RFC8231], a PCC reports the state of its LSPs to the PCE using PCRpt messages and as per [RFC8281], PCE would initiate any missing LSPs and/or remove any LSPs that are not wanted. The same PCEP messages and procedure is also used for the Central Controllers Instructions synchronization. The PCRpt message includes the CCI and the LSP object to report the label forwarding instructions. The PCE would further remove any unwanted instructions or initiate any missing instructions.

5.4.7. PCECC LSP State Report

As mentioned before, an Ingress PCC MAY choose to apply any OAM mechanism to check the status of LSP in the Data plane and MAY further send its status in PCRpt message to the PCE.

6. PCEP messages

As defined in [RFC5440], a PCEP message consists of a common header followed by a variable-length body made of a set of objects that can be either mandatory or optional. An object is said to be mandatory in a PCEP message when the object must be included for the message to be considered valid. For each PCEP message type, a set of rules is

defined that specify the set of objects that the message can carry. An implementation MUST form the PCEP messages using the object ordering specified in this document.

LSP-IDENTIFIERS TLV MUST be included in the LSP object for PCECC LSP.

6.1. The PCInitiate message

The PCInitiate message [RFC8281] can be used to download or remove the labels, the message has been extended as shown below -

```
<PCInitiate Message> ::= <Common Header>
                           <PCE-initiated-lsp-list>
```

Where:

```
<Common Header> is defined in [RFC5440]
```

```
<PCE-initiated-lsp-list> ::= <PCE-initiated-lsp-request>
                              [<PCE-initiated-lsp-list>]
```

```
<PCE-initiated-lsp-request> ::=
    (<PCE-initiated-lsp-instantiation>|
     <PCE-initiated-lsp-deletion>|
     <PCE-initiated-lsp-central-control>)
```

```
<PCE-initiated-lsp-central-control> ::= <SRP>
                                         <LSP>
                                         <cci-list>
```

```
<cci-list> ::= <CCI>
               [<cci-list>]
```

Where:

```
<PCE-initiated-lsp-instantiation> and
<PCE-initiated-lsp-deletion> are as per
[RFC8281].
```

The LSP and SRP object is defined in [RFC8231].

When PCInitiate message is used for central controller's instructions (labels), the SRP, LSP and CCI objects MUST be present. The SRP object is defined in [RFC8231] and if the SRP object is missing, the receiving PCC MUST send a PCErr message with Error-type=6 (Mandatory Object missing) and Error-value=10 (SRP object missing). The LSP object is defined in [RFC8231] and if the LSP object is missing, the receiving PCC MUST send a PCErr message with Error-type=6 (Mandatory Object missing) and Error-value=8 (LSP object missing). The CCI

object is defined in Section 7.3 and if the CCI object is missing, the receiving PCC MUST send a PCErr message with Error-type=6 (Mandatory Object missing) and Error-value=TBD (CCI object missing). More than one CCI object MAY be included in the PCInitiate message for the transit LSR.

To cleanup the SRP object must set the R (remove) bit.

At max two instances of CCI object would be included in case of transit LSR to encode both in-coming and out-going label forwarding instructions. Other instances MUST be ignored.

6.2. The PCRpt message

The PCRpt message can be used to report the labels that were allocated by the PCE, to be used during the state synchronization phase.

```
<PCRpt Message> ::= <Common Header>
                        <state-report-list>
```

Where:

```
<state-report-list> ::= <state-report>[<state-report-list>]
```

```
<state-report> ::= (<lsp-state-report>|
                    <central-control-report>)
```

```
<lsp-state-report> ::= [<SRP>]
                        <LSP>
                        <path>
```

```
<central-control-report> ::= [<SRP>]
                             <LSP>
                             <cci-list>
```

```
<cci-list> ::= <CCI>
               [<cci-list>]
```

Where:

<path> is as per [RFC8231] and the LSP and SRP object are also defined in [RFC8231].

When PCRpt message is used to report the central controller's instructions (labels), the LSP and CCI objects MUST be present. The LSP object is defined in [RFC8231] and if the LSP object is missing, the receiving PCE MUST send a PCErr message with Error-type=6 (Mandatory Object missing) and Error-value=8 (LSP object missing).

The CCI object is defined in Section 7.3 and if the CCI object is missing, the receiving PCC MUST send a PCErr message with Error-type=6 (Mandatory Object missing) and Error-value=TBD (CCI object missing). Two CCI object can be included in the PCRpt message for the transit LSR.

7. PCEP Objects

The PCEP objects defined in this document are compliant with the PCEP object format defined in [RFC5440].

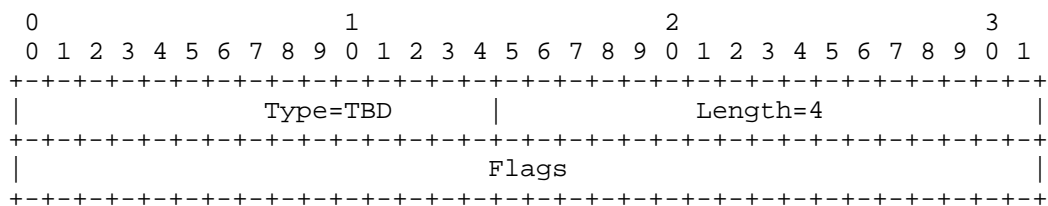
7.1. OPEN Object

This document defines a new optional TLVs for use in the OPEN Object.

7.1.1. PCECC Capability sub-TLV

The PCECC-CAPABILITY sub-TLV is an optional TLV for use in the OPEN Object for PCECC capability advertisement in PATH-SETUP-TYPE-CAPABILITY TLV. Advertisement of the PCECC capability implies support of LSPs that are setup through PCECC as per PCEP extensions defined in this document.

Its format is shown in the following figure:



The type of the TLV is TBD and it has a fixed length of 4 octets.

The value comprises a single field - Flags (32 bits).

No flags are assigned right now.

Unassigned bits are considered reserved. They MUST be set to 0 on transmission and MUST be ignored on receipt.

7.2. PATH-SETUP-TYPE TLV

The PATH-SETUP-TYPE TLV is defined in [I-D.ietf-pce-lsp-setup-type]; this document defines a new PST value:

- o PST = TBD: Path is setup via PCECC mode.

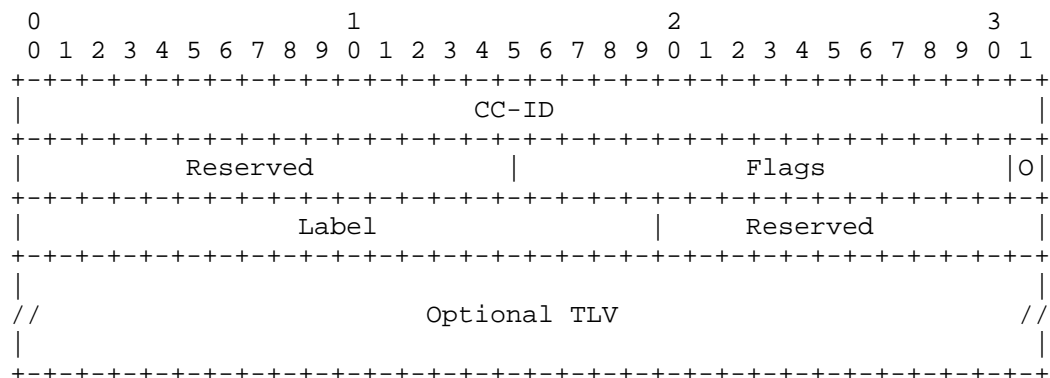
On a PCRpt/PCUpd/PCInitiate message, the PST=TBD in PATH-SETUP-TYPE TLV in SRP object indicates that this LSP was setup via a PCECC based mechanism.

7.3. CCI Object

The Central Control Instructions (CCI) Object is used by the PCE to specify the forwarding instructions (Label information in the context of this document) to the PCC, and MAY be carried within PCInitiate or PCRpt message for label download.

CCI Object-Class is TBD.

CCI Object-Type is 1 for the MPLS Label.



The fields in the CCI object are as follows:

CC-ID: A PCEP-specific identifier for the CCI information. A PCE creates an CC-ID for each instruction, the value is unique within the scope of the PCE and is constant for the lifetime of a PCEP session. The values 0 and 0xFFFFFFFF are reserved and MUST NOT be used.

Flags: is used to carry any additional information pertaining to the CCI. Currently, the following flag bit is defined:

- * O bit(Out-label) : If the bit is set, it specifies the label is the OUT label and it is mandatory to encode the next-hop information (via IPV4-ADDRESS TLV or IPV6-ADDRESS TLV or UNNUMBERED-IPV4-ID-ADDRESS TLV in the CCI object). If the bit is not set, it specifies the label is the IN label and it is optional to encode the local interface information (via IPV4-ADDRESS TLV or IPV6-ADDRESS TLV or UNNUMBERED-IPV4-ID-ADDRESS TLV in the CCI object).

Label (20-bit): The Label information.

Reserved (12 bit): Set to zero while sending, ignored on receive.

7.3.1. Address TLVs

This document defines the following TLVs for the CCI object to associate the next-hop information in case of an outgoing label and local interface information in case of an incoming label.

IPv4-ADDRESS TLV:

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Type=TBD                               |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     IPv4 address                           |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

IPv6-ADDRESS TLV:

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Type=TBD                               |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     //                                     //
|                                     IPv6 address (16 bytes)                 |
|                                     //                                     //
+-----+-----+-----+-----+-----+-----+-----+-----+

```

UNNUMBERED-IPv4-ID-ADDRESS TLV:

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Type=TBD                               |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Node-ID                               |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Interface ID                           |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

The address TLVs are as follows:

IPv4-ADDRESS TLV: an IPv4 address.

IPv6-ADDRESS TLV: an IPv6 address.

UNNUMBERED-IPv4-ID-ADDRESS TLV: a pair of Node ID / Interface ID tuples.

8. Security Considerations

The security considerations described in [RFC8231] and [RFC8281] apply to the extensions described in this document. Additional considerations related to a malicious PCE are introduced.

8.1. Malicious PCE

PCE has complete control over PCC to update the labels and can cause the LSP's to behave inappropriate and cause cause major impact to the network. As a general precaution, it is RECOMMENDED that these PCEP extensions only be activated on authenticated and encrypted sessions across PCEs and PCCs belonging to the same administrative authority, using Transport Layer Security (TLS) [RFC8253], as per the recommendations and best current practices in [RFC7525].

9. Manageability Considerations

9.1. Control of Function and Policy

A PCE or PCC implementation SHOULD allow to configure to enable/disable PCECC capability as a global configuration.

9.2. Information and Data Models

[RFC7420] describes the PCEP MIB, this MIB can be extended to get the PCECC capability status.

The PCEP YANG module [I-D.ietf-pce-pcep-yang] could be extended to enable/disable PCECC capability.

9.3. Liveness Detection and Monitoring

Mechanisms defined in this document do not imply any new liveness detection and monitoring requirements in addition to those already listed in [RFC5440].

9.4. Verify Correct Operations

Mechanisms defined in this document do not imply any new operation verification requirements in addition to those already listed in [RFC5440] and [RFC8231].

9.5. Requirements On Other Protocols

PCEP extensions defined in this document do not put new requirements on other protocols.

9.6. Impact On Network Operations

PCEP extensions defined in this document do not put new requirements on network operations.

10. IANA Considerations

10.1. PCEP TLV Type Indicators

IANA is requested to confirm the early allocation of the following TLV Type Indicator values within the "PCEP TLV Type Indicators" sub-registry of the PCEP Numbers registry, and to update the reference in the registry to point to this document, when it is an RFC:

Value	Meaning	Reference
TBD	PCECC-CAPABILITY	This document
TBD	IPV4-ADDRESS TLV	This document
TBD	IPV6-ADDRESS TLV	This document
TBD	UNNUMBERED-IPV4-ID-ADDRESS TLV	This document

10.2. New Path Setup Type Registry

IANA is requested to allocate new PST Field in PATH- SETUP-TYPE TLV. The allocation policy for this new registry should be by IETF Consensus. The new registry should contain the following value:

Value	Description	Reference
TBD	Traffic engineering path is setup using PCECC mode	This document

10.3. PCEP Object

IANA is requested to allocate new registry for CCI PCEP object.

Object-Class Value	Name	Reference
TBD	CCI Object-Type	This document
	1	MPLS Label

10.4. CCI Object Flag Field

IANA is requested to create a registry to manage the Flag field of the CCI object.

One bit to be defined for the CCI Object flag field in this document:

Codespace of the Flag field (CCI Object)

Bit	Description	Reference
7	Specifies label is out label	This document

10.5. PCEP-Error Object

IANA is requested to allocate new error types and error values within the "PCEP-ERROR Object Error Types and Values" sub-registry of the PCEP Numbers registry for the following errors:

Error-Type	Meaning	
-----	-----	
19	Invalid operation.	
	Error-value = TBD :	Attempted PCECC operations when PCECC capability was not advertised
	Error-value = TBD :	Stateful PCE capability was not advertised
	Error-value = TBD :	Unknown Label
6	Mandatory Object missing.	
	Error-value = TBD :	CCI object missing
TBD	PCECC failure.	
	Error-value = TBD :	Label out of range.
	Error-value = TBD :	Instruction failed.

11. Acknowledgments

We would like to thank Robert Tao, Changjing Yan, Tieying Huang and Avantika for their useful comments and suggestions.

12. References

12.1. Normative References

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

- [RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <<https://www.rfc-editor.org/info/rfc5440>>.
- [RFC7420] Koushik, A., Stephan, E., Zhao, Q., King, D., and J. Hardwick, "Path Computation Element Communication Protocol (PCEP) Management Information Base (MIB) Module", RFC 7420, DOI 10.17487/RFC7420, December 2014, <<https://www.rfc-editor.org/info/rfc7420>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8231] Crabbe, E., Minei, I., Medved, J., and R. Varga, "Path Computation Element Communication Protocol (PCEP) Extensions for Stateful PCE", RFC 8231, DOI 10.17487/RFC8231, September 2017, <<https://www.rfc-editor.org/info/rfc8231>>.
- [RFC8233] Dhody, D., Wu, Q., Manral, V., Ali, Z., and K. Kumaki, "Extensions to the Path Computation Element Communication Protocol (PCEP) to Compute Service-Aware Label Switched Paths (LSPs)", RFC 8233, DOI 10.17487/RFC8233, September 2017, <<https://www.rfc-editor.org/info/rfc8233>>.
- [RFC8281] Crabbe, E., Minei, I., Sivabalan, S., and R. Varga, "Path Computation Element Communication Protocol (PCEP) Extensions for PCE-Initiated LSP Setup in a Stateful PCE Model", RFC 8281, DOI 10.17487/RFC8281, December 2017, <<https://www.rfc-editor.org/info/rfc8281>>.

12.2. Informative References

- [RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <<https://www.rfc-editor.org/info/rfc4655>>.

- [RFC7025] Otani, T., Ogaki, K., Caviglia, D., Zhang, F., and C. Margaria, "Requirements for GMPLS Applications of PCE", RFC 7025, DOI 10.17487/RFC7025, September 2013, <<https://www.rfc-editor.org/info/rfc7025>>.
- [RFC7399] Farrel, A. and D. King, "Unanswered Questions in the Path Computation Element Architecture", RFC 7399, DOI 10.17487/RFC7399, October 2014, <<https://www.rfc-editor.org/info/rfc7399>>.
- [RFC7491] King, D. and A. Farrel, "A PCE-Based Architecture for Application-Based Network Operations", RFC 7491, DOI 10.17487/RFC7491, March 2015, <<https://www.rfc-editor.org/info/rfc7491>>.
- [RFC8253] Lopez, D., Gonzalez de Dios, O., Wu, Q., and D. Dhody, "PCEPS: Usage of TLS to Provide a Secure Transport for the Path Computation Element Communication Protocol (PCEP)", RFC 8253, DOI 10.17487/RFC8253, October 2017, <<https://www.rfc-editor.org/info/rfc8253>>.
- [RFC8283] Farrel, A., Ed., Zhao, Q., Ed., Li, Z., and C. Zhou, "An Architecture for Use of PCE and the PCE Communication Protocol (PCEP) in a Network with Central Control", RFC 8283, DOI 10.17487/RFC8283, December 2017, <<https://www.rfc-editor.org/info/rfc8283>>.
- [I-D.ietf-teas-pcecc-use-cases]
Zhao, Q., Li, Z., Khasanov, B., Ke, Z., Fang, L., Zhou, C., Communications, T., and A. Rachitskiy, "The Use Cases for Using PCE as the Central Controller(PCECC) of LSPs", draft-ietf-teas-pcecc-use-cases-01 (work in progress), May 2017.
- [I-D.ietf-pce-lsp-setup-type]
Sivabalan, S., Tantsura, J., Minei, I., Varga, R., and J. Hardwick, "Conveying path setup type in PCEP messages", draft-ietf-pce-lsp-setup-type-10 (work in progress), May 2018.
- [I-D.ietf-pce-pcep-yang]
Dhody, D., Hardwick, J., Beeram, V., and J. Tantsura, "A YANG Data Model for Path Computation Element Communications Protocol (PCEP)", draft-ietf-pce-pcep-yang-07 (work in progress), March 2018.

[I-D.zhao-pce-pcep-extension-pce-controller-sr]

Zhao, Q., Li, Z., Dhody, D., Karunanithi, S., Farrel, A.,
and C. Zhou, "PCEP Procedures and Protocol Extensions for
Using PCE as a Central Controller (PCECC) of SR-LSPs",
draft-zhao-pce-pcep-extension-pce-controller-sr-02 (work
in progress), March 2018.

Appendix A. Contributor Addresses

Udayasree Palle
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560066
India

EMail: udayasreereddy@gmail.com

Mahendra Singh Negi
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560066
India

EMail: mahendrasingh@huawei.com

Katherine Zhao
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95050
USA

EMail: katherine.zhao@huawei.com

Boris Zhang
Telus Ltd.
Toronto
Canada

EMail: boris.zhang@telus.com

Authors' Addresses

Quintin Zhao
Huawei Technologies
125 Nagog Technology Park
Acton, MA 01719
USA

EMail: quintin.zhao@huawei.com

Zhenbin Li
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

EMail: lizhenbin@huawei.com

Dhruv Dhody
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560066
India

EMail: dhruv.ietf@gmail.com

Satish Karunanithi
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560066
India

EMail: satishk@huawei.com

Adrian Farrel
Juniper Networks, Inc
UK

EMail: adrian@olddog.co.uk

Chao Zhou
Cisco Systems

EMail: choa.zhou@cisco.com