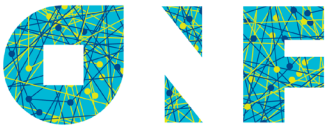


OPEN NETWORKING
FOUNDATION

OpenFlow Controller Benchmarking Methodologies

Version 1.0
November 2016

ONF TR-539

ONF Document Type: Technical Recommendations
ONF Document Name: OpenFlow Controller Benchmarking Methodologies

Disclaimer

Open Networking Foundation
2275 E. Bayshore Road, Suite 103, Palo Alto, CA 94303
www.opennetworking.org

Table of Contents

List of Figures

List of Tables

I. Introduction

One of the key promises of Software Defined Networking (SDN) is to lay the foundation for a more flexible and adaptable network. It achieves the flexibility by separating out the control plane and data plane as independent entities. The data plane remains in the forwarding device and the control plane moves out as an independent software entity. The desired adaptability is achieved by making the network programmable at the software level. Separating control from the data also brings in simpler ways of managing the network as a whole as opposed to managing an individual device in isolation. Programmability at the software level allows network engineers to develop purpose built, tailor made software applications on top of the network operating system.

Scope and Objectives

OpenFlow is an implementation of the above-mentioned concepts. The controller represents the network operating system. It provides north bound API for application development. The controller becomes the central and key component of an OpenFlow network. The operational behavior and efficiency of the controller is a significantly influencing factor for the Software Defined Network. Hence it is important to understand the factors influencing the controller performance and the key metrics a controller needs to satisfy. An understanding of these factors and metrics is essential for planning an OpenFlow based deployment.

OpenFlow Switch Specification 1.3.4 [3] has been followed as the reference version for the methodologies of this document, however these methodologies apply to any latter version of the specification published so far (June 2016).

Common Terms, Abbreviations and Definitions

The terminologies applicable to this OpenFlow benchmarking proposal are:

- **Controller** – The OpenFlow Controller that uses the OpenFlow protocol to populate the flow table of the connected switch.
- **DUT** – The Device Under Test (DUT), which is this OpenFlow Controller.
- **Switch** – The OpenFlow switch that establishes an OpenFlow (OF-Channel) connection to an OpenFlow controller and supports the required OpenFlow messages.
- **Hosts** – The hosts that are connected to the OF switches. These hosts are used by source/destination for data traffic.
- **User** – The user evaluating the OpenFlow Controller.
- **Flap** – When an established channel goes down and comes up, it is termed as a session flap.
- **Flow Setup** – A controller sends flow add messages to the switch to set up a new flow or modify an existing flow

II. Influencing factors for an OF controller

This section describes the background behind the benchmarking methodologies described in this document.

A controller must be connected to multiple switches in the network through secured or unsecured connections. These connections are referred as channels. A controller's workload is influenced by the number of channels it is handling simultaneously and by the operations performed over those channels. While planning for a SDN based network deployment, it is required to know the OF controller's capacity of channel handling. A thorough benchmarking of the channel is needed. It is important to know how many simultaneous channels a controller can support and the rate at which the controller is able to establish connection with the switches. If the channel establishment rate is too slow, then the whole network starts slowly. Too slow a network start up time may not be desirable in certain cases. Optionally, the controller can establish multiple channels with a switch (primary and multiple auxiliaries). For smooth operation, these channels must be equally efficient in generating responses to the messages coming from the switch. Communication of one channel should not negatively influence the operational efficiency (For example, Packet_in processing rate, flow_mod generation time, and so on.) of another channel. Finally, to achieve reliability and redundancy, multiple controllers are used and each controller works in master or slave mode.

For a controller, this document proposes to benchmark the controller channel in the following four areas:

- Benchmark for Channel Capacity and Channel Establishment Rate
- Benchmark for Packet Out Transmission Efficiency of Auxiliary Channel (optional feature)
- Benchmark for Master-Slave Recovery (optional feature)
- Benchmark for Role Initiation Time by Multiple Controllers to a network of switches

An OpenFlow controller programs the network by setting up flows for the switches. Controllers can proactively install some flows anticipating future need or it can install the flows in reactive mode, that is, only when an unknown type of packet arrives at the network. Whenever any new type of packet arrives to a switch, resulting in a table miss, the switch can be programmed to send the packet to the controller using a packet_in message. The controller reacts to the unknown packet arrival and intelligently sets appropriate flows for the packet to traverse through the network. Controller performance and robustness is heavily influenced by processing packet_in(s). To achieve faster network convergence and network recovery, it is important to measure and know the efficiency of a controller dealing with various packet_in(s) and push down flows, under various conditions.

This document proposes the following benchmarking methodology related to the flow modifications (primarily add) and uses various types of packet_in(s) to do the benchmark.

- Controller - Flow Setup Benchmark:
    - Flow Setup Rate: Proactive Mode
    - Flow Setup Rate: Reactive Mode
    - Flow Setup Rate over Auxiliary Channels: Reactive Mode
    - End-to-End Flow Install Time

- Controller Response Time / Latency

OpenFlow Controllers must be intelligent and capable enough to recover from a faulty network condition. OpenFlow Switch specification does provide various mechanisms to do that, for example, Fast Failover. At the same time controllers must be capable enough to detect when the network condition improves and utilize this knowledge to make better operational decision. For example move traffic through better path when a better path is available anytime in future. This document talks about methodologies to measure the effectiveness of a controller in a topology failover situation and also in a topology improved situation (for example better links are available for use now.).

- Benchmark for – Path Recovery Time in "Topology Failure" Condition.
- Benchmark for – Path Switchover Time in a "Topology Improvement" Condition.

A functional controller must be aware of the topology of the switch network and create packet traversal path in the network. One of the suggested ways to discover the underlying network topology is through the propagation of LLDP packets sent from the Controller through packet_out messages to each switch. The switch floods it out through all interfaces and returns received LLDP packets to the controller through a packet_in messages. The faster a controller discovers any given topology, the quicker it will be in a position to create end-to-end path between source and destination for data transfer. This document proposes a methodology to benchmark the topology discovery time and how fast a controller reacts when any change (positive or negative) in the existing topology is detected through LLDP.

- Benchmark for – Recovery Time for Link Connectivity Failure (Detected Through LLDP)
- Benchmark for – Switchover Time for Link Connectivity Improvement (Detected Through LLDP)

The rest of the document provides the details for each and every methodology described briefly above.

1 Test 1: OpenFlow Controller Channel Capacity

1.1 Objective

The purpose of this test is to find out the maximum number of OpenFlow channels that a controller can support simultaneously. This test is done with a single controller having direct TCP connection (no IP hop) established with multiple OF switches. If the OF switches used for testing the controller are physical switches, then there should be a L2 switch between the controller port and the physical OF switches. If the OF switches are simulated by a test tool, then the controller and the test tool should be directly connected with each other.

1.2 Test setup

1.2.1 Topology

The OF-Channel between the controller and switch is based on TCP. The controller interface and the switch interfaces should belong to the same subnet to minimize complexity of the test setup.

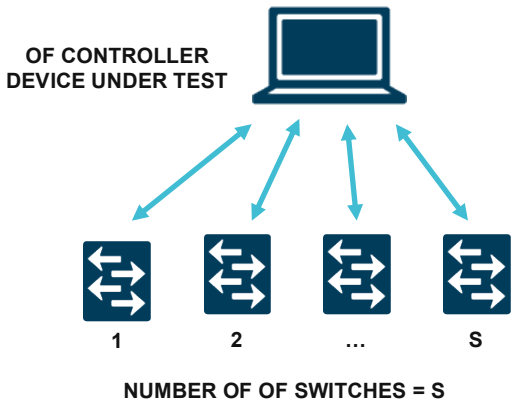


Figure 1.1: Test Topology for Channel Capacity Test

1.2.2 Prerequisites and Recommendations for the test

- Connect the controller under test to the OF switches using the minimum number of IP hops necessary, ideally zero, so that the possibility of error conditions due to the other network activity and congestion are minimized.
- Use same switch simulator or real switches when testing with different controllers, so that switch side latencies remain a common factor for benchmarking different controllers from different vendors.

1.3 Test configuration

1.3.1 Controller configuration

Set the following controller parameters before proceeding with this test case.

Table 1.1: Controller Configuration Parameters

Parameter	Comments
Channel Type	TCP or TLS
Hello Request	Enable sending Hello Request to each switch once the OF channel is established.
Feature Request	Enable sending Feature Request message once the OF channel with switch has come up (mandatory if Hello is not enabled).
Echo Reply	Should reply to Echo Request sent by each switch.
Flow Profile	There are no preconfigured flows in the controller.
Topology Discovery	Disabled

1.3.2 Switch configuration

Set the following switch parameters before proceeding with the test case.

Table 1.2: Switch Configuration Parameters

Parameter	Comments
Channel Type	TCP or TLS as configured in controller.
Echo Request	Enable sending periodic Echo Request to controller once the OF channel is established.

1.4 Test steps

1. Start the controller.
2. Start capturing on the controller interface through which it is connected to all the switches. If the switches are simulated by a tool, then the capture can be started at tool port connected to controller.
3. Start switches that will establish the OF channels with the controller
4. Wait for a user configured time duration by which all the *S* number of channels are established. Let that be $T_{ramp\text{-}up}$. If all the channels are not established within $T_{ramp\text{-}up}$, then restart the test with a lesser number of channels.
5. Once all the channels are up, verify that the controller has sent out at least one Hello message or Feature Request message to each of the *S* switches. If this check fails, then restart the test with a lesser value of *S. (use binary search and deduce the value of S for next iteration)*.
6. If step 5 is successful, then wait for a user configured time period and check whether all the channels remain in established state without any channel flap (channel going down and up is termed as a Channel Flap). This waiting period is referred as $T_{stabilize}$. $T_{stabilize}$ is user defined, recommended value is 5 min.

7. To check whether each session is up and healthy, each switch sends periodic echo requests to the controller with a one-minute frequency (it is recommended to have a reasonable frequency such that the controller is not stuck by very fast packet processing, resulting in session going down). Recommended value is one Echo Request per min.
8. Now check whether all the Echo Requests sent out by the switches are replied by the controller without any miss. Number of Echo Reply messages transmitted by the controller must be same as number of Echo Request messages sent by the switches. Any mismatch in numbers indicate that the controller is not able to sustain S number of OpenFlow connections.
9. If the above check succeeds, then restart the test with higher number of switches; else restart the test with lesser value of *S*. The value to be tried is determined by the logic of binary search or a preferred algorithm to step up or step down the number of switches in case of success or failure respectively.
10. Find out the maximum value of S following the above mentioned process.

1.5 Test verification

Once all the channels are established, the following conditions must be met:

- Either Hello or Feature Request messages must be sent out to all the switches from the controller.
- Channels must remain in established state without any flap for $T_{stabilize}$.
- All Echo Request messages sent out by the switches must be replied by the controller.

1.6 Test measurement

The following image depicts the time graph of the test that represents the test in a scale of time:

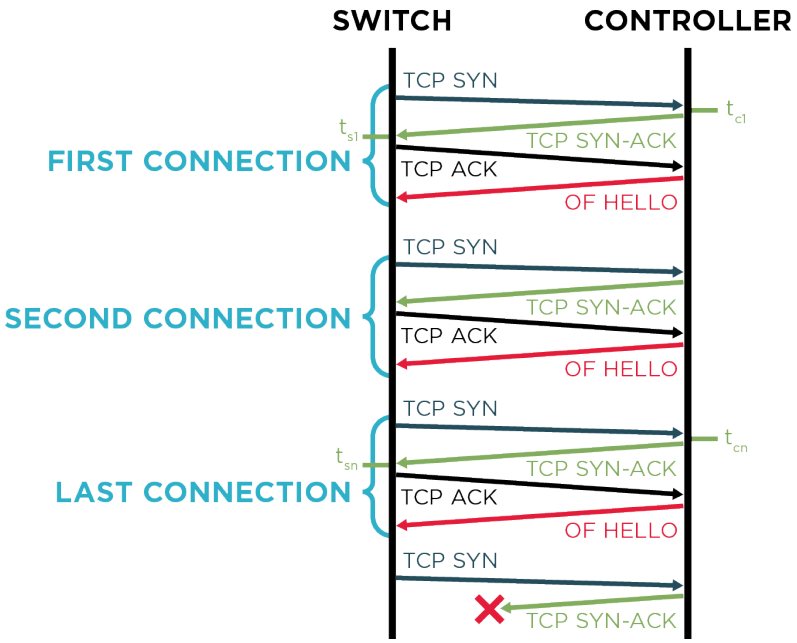


Figure 1.2: Test Diagram of Channel Capacity Test

Repeat the test for N iteration to find out the average channel capacity calculated as:

Average Channel Capacity = $\sum_{i=1}^{N} Si/N$, where Si is the channel capacity in i-th iteration.

1.7 Test Iterations: Combination of Variables

The following variables might affect the outcome of the test and the test can be iterated with different combinations of such variables.

Table 1.3: Variable Combinations

Parameters	**Possible Options**
Channel Type	TCP or TLS
Periodic Echo Tx at Controller	Enabled or Disabled (If enabled at what frequency)
Feature Request Tx by Controller	Enabled or Disabled
Hello Tx by Controller	Enabled or Disabled
Statistics Request by Controller	Enable or Disable specific stat requests as part of the channel establishment

1.8 Test report format

Test result of a single iteration to find out the maximum channel capacity can be represented in following tabular format.

Table 1.4: Channel Capacity Test Report

Periodic Hello Enabled	Feature Request Enabled	Echo Request Enabled	LLDP Enabled	Channel Type	Number of Channels	Channel Establishment Time(seconds)	Result
Yes	yes	yes	No	tcp	400	14	Pass
Yes	yes	yes	No	tcp	512	25	Pass
Yes	yes	yes	No	tcp	1024	301	Fail
Yes	yes	yes	No	tcp	768	24	Pass
Yes	yes	yes	No	tcp	896	63	Pass
Yes	yes	yes	No	tcp	960	32	Pass
Yes	yes	yes	No	tcp	992	31	Pass
Yes	yes	yes	No	tcp	1008	26	Pass

In the example above, the maximum channel capacity achieved is 1008. The test can be reiterated with some different combination of the highlighted parameters for channel type TCP.

2 Test 2: Measure Packet Out Transmission Efficiency of Auxiliary Channel

2.1 Objective

This test is based on Openflow specification 1.3.4 sec 6.3.6. According to the specification, all packet_out messages containing a data packet from a packet_in message should be sent on the same connection where the packet_in came in. There is no synchronization between connections, and messages sent on different connections may be processed in any order. Therefore, the controller's utilization of auxiliary channels is controlled by the switch. The purpose of this test is to find out, when switch initiates auxiliary channel and performs message exchanges and then how it impacts controller's efficiency as opposed to doing interaction over a single primary channel. In this test, a Switch generates a large number of packet_in messages over different auxiliary channels and the controller has to respond to those large number of packet_in messages received over specific auxiliary channels with packet_out messages. This test will measure that when each auxiliary channel is loaded with equal load (in coming packet_in to the controller); then for each of the auxiliary channel, the response rate (packet_out) is also equal or near equal.

2.2 Test Setup

2.2.1 Topology

A single Switch is connected to a single controller having a primary channel and N number of Auxiliary channels. The connection may be TCP or UDP. The switch has two ports connected to two Hosts, port 1 and 2.

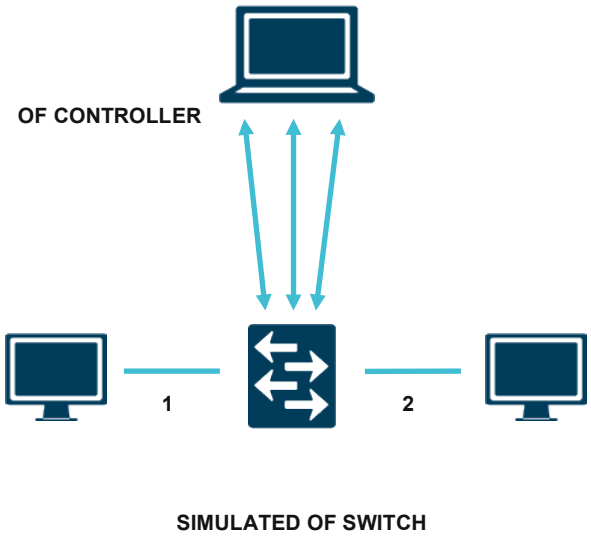


Figure 2.1: Topology for Auxiliary Channel Efficiency

2.2.2 Prerequisites and Recommendations for the test

- The controller is directly connected (no additional IP hops) to the switch (simulated by the test tool) to remove any error condition due to the other network activity and congestion.
- Use same switch simulator or real switches when testing with different controllers, so that switch side latencies remain a common factor for benchmarking different controllers from different vendors.
- Both the controller and Switch should support Auxiliary channels.
- Controller should respond to packet_in messages with packet_out messages over the same channel where the packet_in was received.

2.3 Test Configuration

2.3.1 Controller Configuration

The controller must be configured with parameters given in Table 2.1 to meet the objective of the test. Other configuration parameters must be kept at default. A few example iterations are defined in this document in Table 2.1 below, but there can be multiple iterations with different combination of these parameters. For this test, it is assumed that the controller will reply with flow_mod message for each packet_in that it receives and it is capable of using Auxiliary channels to transmit flow_mod messages.

Table 2.1: Controller Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2
Auxiliary Channel Type	TCP or UDP	TCP	UDP
Hello Element	Optional parameter	Enabled.	Disabled
Multipart Request	Optional parameter	Enabled.	Enabled
Echo Request/Reply	Optional parameter	Enabled.	Enabled
Barrier Request	Optional parameter	Enabled	Enabled
Number of Auxiliary channels	Number of Auxiliary channels between the Switch and controller	N1	N1
Controller Application to respond packet_in	Should run application that responds to packet_in messages with flow_mod messages	Enabled	Enabled

2.3.2 Switch Configuration

On the test tools side, following switch parameters must be set before proceeding with the test case. Few example set of values for iteration are illustrated below, but there can be different combinations over which the test can be iterated.

Table 2.2: Switch Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2
Auxiliary Channel Type	TCP or UDP	TCP	UDP
Hello message	Switch must be able to reply to Hello from controller.	Enabled	Enabled
Multipart Reply	Must be able to reply to Multipart Request from the controller.	Enabled	Enabled
Echo Request/Reply	Optional parameter.	Enabled	Enabled
Barrier Reply	Must reply to Barrier Request received from controller.	Enabled	Enabled
Number of Auxiliary Channels	Total number of Auxiliary Channels.	N1	N2
Number of packet_in messages	Number of packet_in messages that the switch will send out once the OF channel is established with the controller	10k/switch	10k/switch

2.4 Test Steps

1. Configure the Switch such that it sends large number of packet_in messages with unique L2/L3 header and equally distributed between the Auxiliary channels (For example, 10k per Auxiliary channel).
2. In each auxiliary channel, there must be two sets of packet_in messages: set 1 where the in_port should be 1 for source mac X to destination mac Y and set 2 where the in_port should be 2 for source mac Y and destination mac X. The same configuration applies for IP addresses of L3 profiles. Bi-directionality is required to enable controller know about the location of both source and destination Hosts and push proper flows.
3. Start packet capture on the switch side.
4. Start controller followed by the Switch.
5. Wait till the switch have sent out all the packet_in messages configured. This can be verified by packet_in tx statistics on the switch.
6. Wait till the controller has replied to all the packet_in messages received. Can be verified by looking at packet_out Tx counter in the Controller.
7. Stop capture.
8. Look out for feature reply messages from the switch over each Auxiliary channel. Note down the Auxiliary Id mentioned in each of the feature reply and the corresponding source TCP/UDP port of that message. This TCP/UDP port must be used for identifying each of the Auxiliary channels.
9. Measure the packet_out rate over each Auxiliary channel.
10. Re-iterate the test with different combination of parameter values.

2.5 Test Measurement

1. Filtering by the destination TCP/UDP port, find out the following:

a. Number of packet_out the controller has sent through a particular Auxiliary channel, for example, consider N1 (assuming controller may not use the same Auxiliary channel to send packet_out for all 10k packet_in messages received over that Auxiliary channel).
b. Measure the time it took to send those N1 packet_out, let that be T1
c. Find out the rate as Packet_out_tx_rate_1 $= N1/T1$

2. Measure above parameters for each Auxiliary channel and then compare.

2.6 Test Results

Here is an example of presenting the data in tabular form (Auxiliary Id 0 signifies the primary channel):

Table 2.3: Result Format

Type of Channel	Hello Message	Barrier Request	Periodic Echo	Periodic Multipart Request	Auxiliary Id	No of packet_out	Packet_out Tx rate
TCP	Enabled	Enabled	Disabled	Disabled	0	N	F/sec
					A1	N1	F1/sec
					A2	N2	F2/sec
					A3	N3	F3/sec

2.7 Test Iterations: Combination of Variables

If the number of Auxiliary channels is kept constant, the following variables might affect the outcome of the test and can be iterated over with different combinations.

Table 2.4: Iterations Format

Parameters	Possible Options
Channel Type	TCP or UDP
Hello Message	Enabled/Disabled
Barrier Request	Enabled/Disabled
Echo Request	Enabled/Disabled
Type of Flow Profile	L2, IP, TCP

3 Test 3: Measure Master-Slave Recovery Time

3.1 Objective

The purpose of this test is to find out the time taken for a cluster of controllers to elect a new master when the current master fails. This time can be measured by sending continuous packet_in messages from the switch and then finding out the time duration between the last responded packet_in message by the previous Master and the first role_request message from the newly elected Master. This test can be done using Echo Request & Echo Reply as well. In that case, packet_in will be replaced by Echo Request and packet_out to be replaced by Echo Reply messages as described below. The usage of packet_in Vs. Echo Request is end user's discretion.

3.2 Test Setup

3.2.1 Topology

A single switch is connected to a controller cluster of multiple controllers. The connection type may be TCP or TLS. The controllers and the switch all are connected to the same LAN.

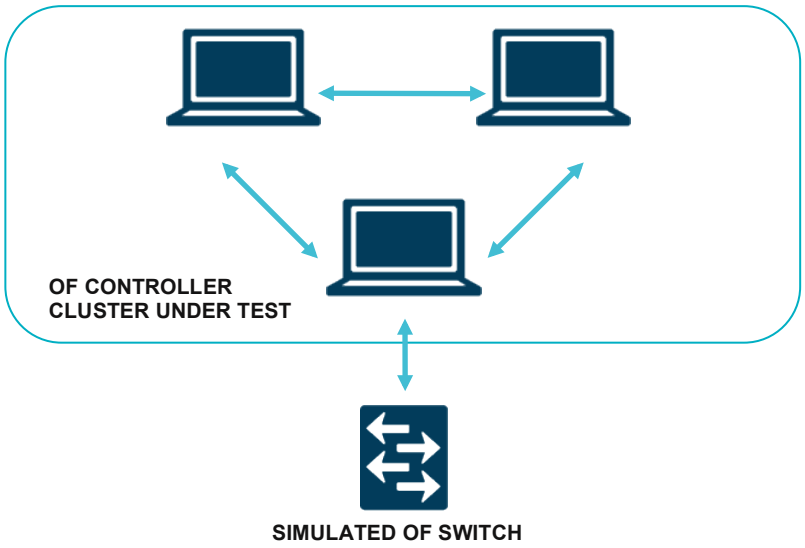


Figure 3.1: Test Topology for Master-Slave Recovery Time

3.2.2 Prerequisites and Recommendations for the test

- The controller is directly connected (no additional IP hops) to the switch (simulated by the test tool) to remove any error condition due to the other network activity and congestion.
- Use the same switch simulator or real switches when testing with different controllers, so that switch side latencies remain a common factor for benchmarking different controllers from different vendors. The controller cluster also need to maintain the same topology within the cluster to eliminate variations, when test is repeated for different controllers coming from different vendors.
- Controllers should be able to send packet_out or flow_mod message in response to packet_in message with encapsulated ARP Request header. The test can be done with any type of header (in place of ARP) but the choice of header needs to be consistent when testing across controllers.
- The Controller may reply with flow_mod message in response to packet_in from the switch. In that case all measurements related to packet_out message should be done for flow_mod message.
- The test can also be done with echo_request message instead of packet_in. In that case the Controller must respond with echo_reply message. All measurements taken for packet_in and packet_out message should be taken for echo_request and echo_reply message respectively.

3.3 Test Configuration

3.3.1 Controller Configuration

The controller must be configured with following configuration parameters to meet the objective of the test. Other configuration parameters must be kept at default. As an example, two iterations are defined in the table below.

Table 3.1: Controller Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2
Channel Type	Type of channel established between controller and Switch.	TCP	TLS
Support multiple controller interaction	Should be able to interact with multiple controllers and take part in Master election.	Enabled	Enabled
Response to packet_in	Controller should send packet_out or flow_mod message for each packet_in message received	Enabled	Enabled
Echo_Reply	Controller should send echo_reply message when it receives echo_request from the switch.	Enabled	Enabled

3.3.2 Switch Configuration

The following Switch parameters must be set before proceeding with the test case. Parameters for two iterations are illustrated below.

Table 3.2: Switch Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2
Channel Type	Type of channel established between controller and Switch.	TCP	TLS
Support Multiple Controllers	Each switch should be able to establish OF channel with multiple controllers.	Enabled	Enabled
Packet_in Tx	Switch must be configured to send large scale packet_in messages with unique message header. The suggested message header for this test is ARP. The switch should be able to send packet_in with unique ARP packet encapsulated.	Enabled	Enabled
Echo_req Tx	Switch should be able to generate echo_req message with lowest possible interval	Enabled	Enabled

3.4 Test Steps

1. Configure one controller in the cluster as master and the rest as slaves.
2. Configure the switch such that the switch is capable of sending packet_in messages with ARP Request (it can be something else, but this inner header must be consistent throughout the testing, across various controllers) as header type, with a high frequency for example 1k/sec.
3. The time period, at the end of which the test tool send packet_in, let us call it T_{pktin_intrvl}
4. Start packet capture on the switch side.
5. Start all controllers
6. Start all switches.
7. Wait till the switch starts sending packet_in and the master controller starts replying with packet_out messages.
8. The average time difference between a packet_in sent and its corresponding packet_out received, is referred as $T_{avg\_resp\_intrvl}$ throughout this test.
9. Once the packet_out receive rate $T_{avg\_resp\_intrvl}$ has stabilized, then bring down the Master controller.
10. Wait till a new Master comes up and sends role_request message with Master role to all the switches for which it was Slave.
11. Stop capture.
12. Measure the recovery time as mentioned in the section 3.5.
13. Iterate as necessary with different combinations.

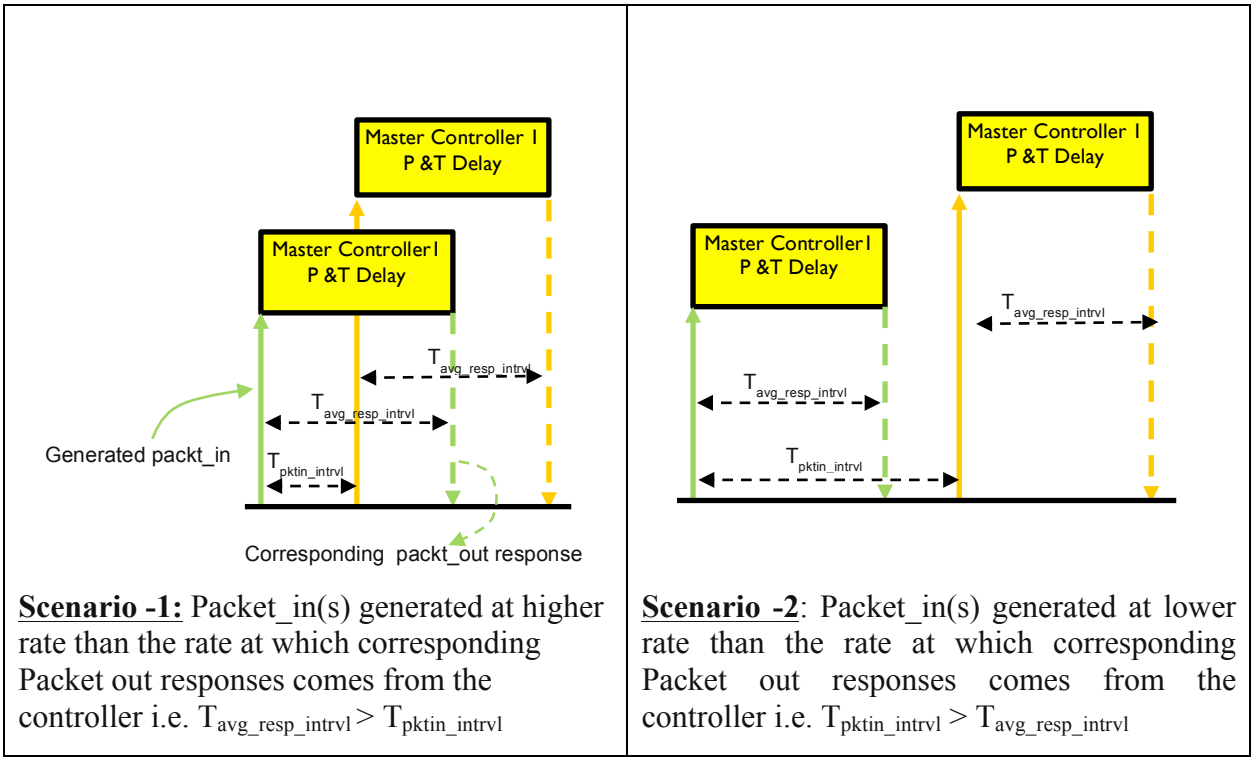
3.5 Test Measurement

Measurement Scenarios:-

In the previous section we have defined two parameters to be measured. The first one is T_{pktin_intrvl} i.e. every T_{pktin_intrvl} time interval one packet_in is generated by the test tool. The second parameter to be measured is $T_{avg\_resp\_intrvl}$. This represents the average time the controller takes to respond to a packet_in. In this test the controller responds with a packet_out against each and every incoming packet_in.

Based on the test configuration and the performance of the controller we can face two scenarios. In one scenario the controller will take more time to respond than the rate at which the incoming packet_in is received i.e. $T_{avg\_resp\_intrvl} > T_{pktin\_intrvl.}$ The second scenario is where the controller is responding at faster or same rate as the incoming packet_in i.e. $T_{pktin\_intrvl} >= T_{avg\_resp\_intrvl}$

These two scenarios are described pictorially below.



Scenario -1: Packet_in(s) generated at higher rate than the rate at which corresponding Packet out responses comes from the controller i.e. $T_{avg\_resp\_intrvl} > T_{pktin\_intrvl}$

Scenario -2: Packet_in(s) generated at lower rate than the rate at which corresponding Packet out responses comes from the controller i.e. $T_{pktin\_intrvl} > T_{avg\_resp\_intrvl}$

Figure 3.2: Master-Slave Recover Time, Scenarios

Measurement Procedure:

1. Filter on the OpenFlow messages in the capture and find out the average time gap between a packet_in transmitted and the corresponding packet_out received at the very beginning, when the first master was up and running. Let that be $T_{avg\_resp\_intrvl}$ sec.

$$Tavg\_resp\_intrvl = [\sum_{i=1}^{N}(TpktOut(i) - TpktIn(i))]/N$$

2. Now look for the last packet_out from the first master and the first role_request message with role set to "Master" from the newly elected master. Note down the gap, let that be T sec. i.e. the distance between point A and point C in the Figure 3.3: Time Diagram for Master-Slave Recovery Time, Scenario 1.

The Recovery Time calculation procedure is explained below (if the test is done with echo_request/echo_reply, then replace packet_in by echo_request and packet_out by echo_reply in the description below):

In this measurement procedure we are faced with an uncertainty while trying to capture the exact moment when the old master went down. Let us call this uncertainty as Failure Detection Uncertainty (FDU). From the testing end, though the precise measurement of exact time when the first master controller was brought down is not possible, but a reasonably accurate estimate of the same can be made. While the test is going on, the average packet_in response time of the active master controller is calculated continuously, that is, the average time the master controller takes to send a packet_out response for an incoming packet_in. We refer it as $T_{avg\_resp\_intrvl}$. During the test, the master controller responded to packet_in every $T_{avg\_resp\_intrvl}$ time. So the controller must have been brought down any time between the last packet_out response from the master controller and the next $T_{avg\_resp\_intrvl}$ time period ending time. In Figure 3.3: Time Diagram for Master-Slave Recovery Time, Scenario 1 this is the time range between point A and point B. The master controller must have gone down anytime between point A and point B.

Point A is the time when the last packet_out response came from the master controller and this can be calculated very accurately. Let it be $T_{lastPacketOutBeforeMasterDown}$. Also, it is known that when the first packet_in that was sent to the master controller for which switch has not received the packet_out response. As shown in Figure 3.3: Time Diagram for Master-Slave Recovery Time, Scenario 1 (the upward black arrow) let it be $T_{firstUnrespondedPacketIn}$. With these tow measurements, the maximum Failure Detection Uncertainty (FDU) is calculated for scenario-1 (Ref Figure 3.2: Master-Slave Recover Time, Scenarios), where $T_{avg\_resp\_intrvl} > T_{pktin\_intrvl}$ as:

$$FDU_{max} = [Tavg - (TlastPacketOutBeforeMasterDown - TfirstUnrespondedPacketIn)]$$

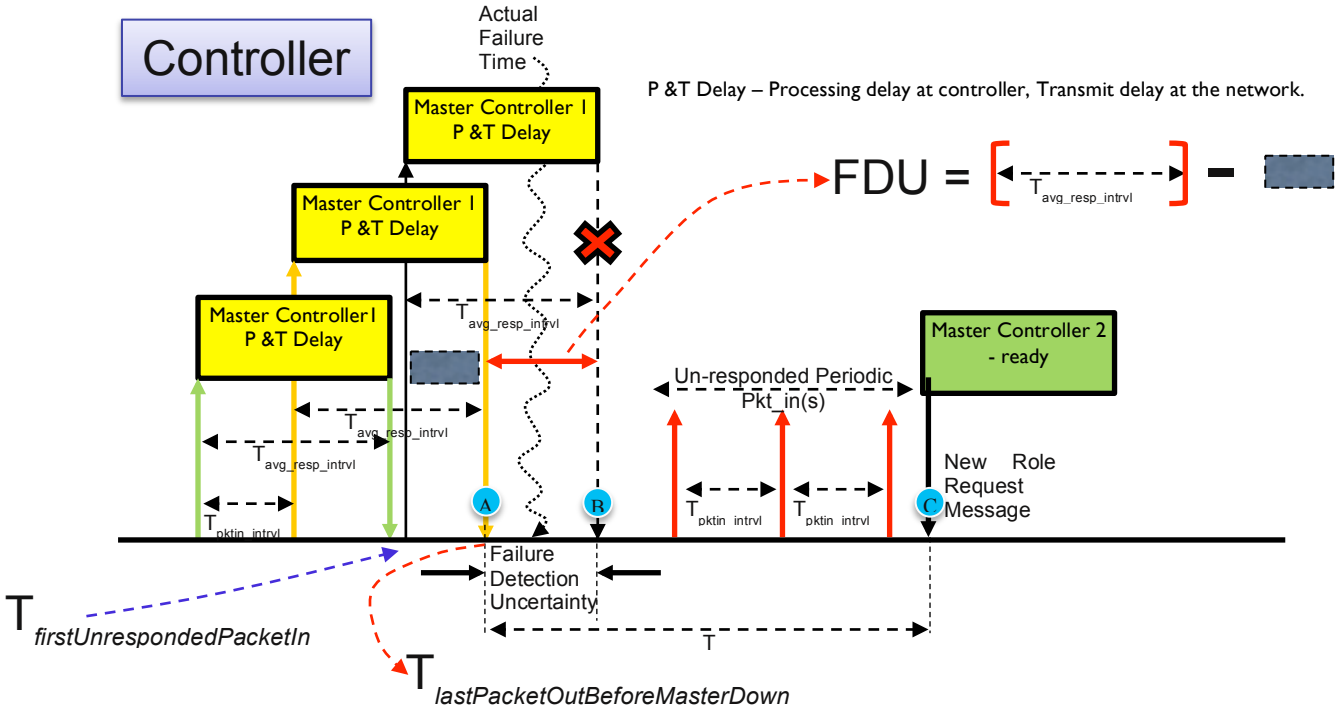


Figure 3.3: Time Diagram for Master-Slave Recovery Time, Scenario 1

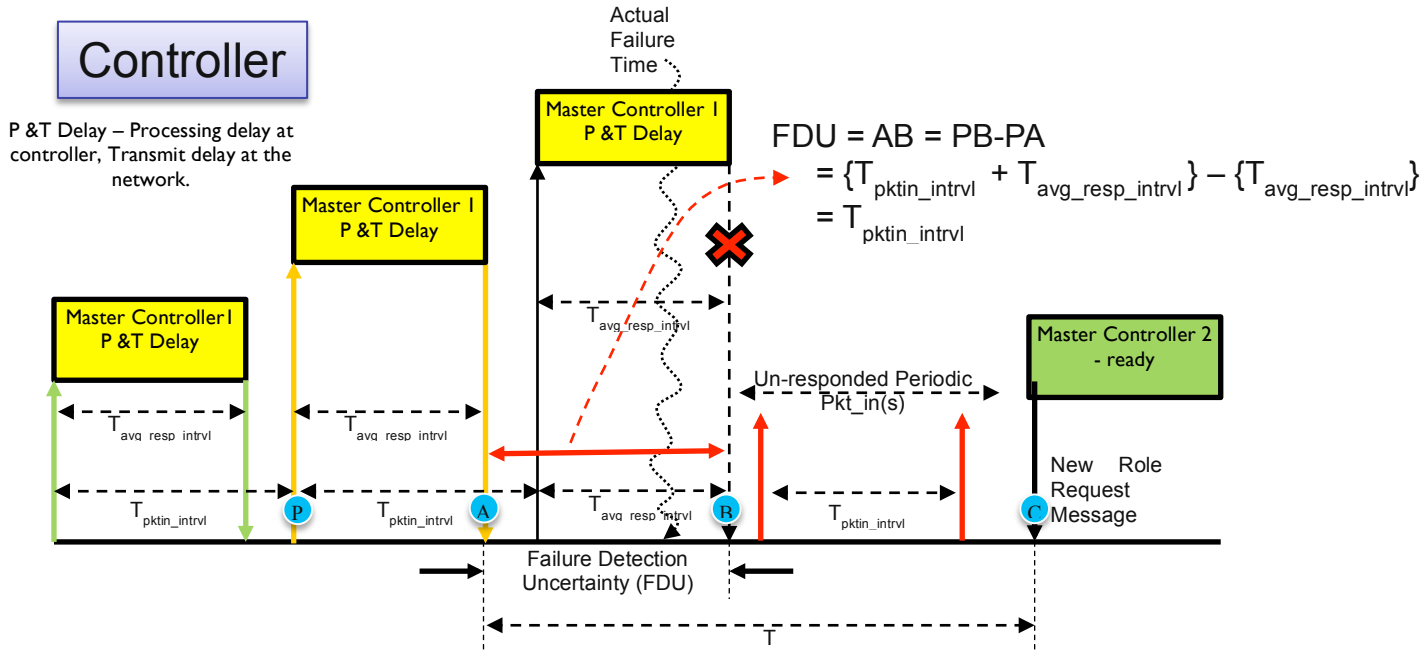


Figure 3.4: Time Diagram for Master-Slave Recovery Time, Scenario 2

For scenario-2 (Ref Figure 3.2: Master-Slave Recover Time, Scenarios), where $T_{pktin\_intrvl} > T_{avg\_resp\_intrvl}$ FDUmax is calculated as:

$$FDU_{max} = T_{pktin\_intrvl}$$

With this accurate estimation of FDUmax, the controller recovery time can be calculated. As illustrated in the Figure 3.3: Time Diagram for Master-Slave Recovery Time, Scenario 1 & Figure 3.4: Time Diagram for Master-Slave Recovery Time, Scenario 2 let T be measured time between the last successful packet_out received from the old master and the time when the first role_request received from the new master.

$$T = (TroleReqFromNewMaster - TlastPacketOutBeforeMasterDown)$$

So recovery time can be calculated as:

$$\text{Recovery Time (min)} = [T - (FDUmax)]$$

$$\text{Recovery Time (max)} = T \text{ , when there is no uncertainty}$$

Observation 1 - It must be noted that the value of FDU_{max} depends upon two factors, $T_{avg\_resp\_intrvl}$ and T_{pktin_intrvl}. The smaller the T_{pktin_intrvl}, smaller will be FDU_{max} if $T_{avg\_resp\_intrvl}$ is less than T_{pktin_intrvl}. For the other case when $T_{avg\_resp\_intrvl}$ is higher, then tester does not have any control to reduce the FDU_{max}, since $T_{avg\_resp\_intrvl}$ depends entirely on controller performance.

Observation 2:- Assume that time T is in the order of few seconds. For the sake of discussion, assume it to be 5 sec. Also assume FDU_{max} is in the range of millisecond. For the discussion, assume it to be 2 milliseconds. In this case, the recovery time will have minimum value of 4.998

sec to a max of 5 sec. So the degree of error introduced for larger value of recovery time is also very small. Thus, the uncertainties can be ignored to keep the testing simple. In this case recovery time is T.

Recovery Time = T , $for\ larger\ value\ of\ T\ in\ socnds$

Based on the above discussion, the formulae for calculating the recovery time are:

1. For Recovery time in the range of milliseconds

   Recovery Time (min) = $[T\ -(FDU_{max})]$

   Recovery Time (max) = T

2. For recovery time in the range of seconds

   Recovery Time = T

3.6 Test Results

Here is an example of presenting the data in tabular form:

Table 3.3: Result Format

Type of Channel	**Number of controllers**	**Recovery Time**
TCP	C1	T1 sec
	C2	T2 sec
	C3	T3 sec

3.7 Test Iterations: Combination of Variables

Following variables might affect the outcome of the test and can be iterated over with different combinations.

Table 3.4: Iterations Format

Parameters	**Possible Options**
Channel Type	TCP or TLS
Number of controllers	C1, C2, C3 etc.

4 Test 4: Measure the Role Intimation Time for a Controller Cluster At Startup

4.1 Objective

Controllers in a cluster need to run an election mechanism among themselves to elect a master. This election mechanism is out of scope for OF specification. If the election mechanism takes too much time, then that may become an efficiency concern. Till a master controller is elected, the SDN network will run without any controller. The purpose of this test is to find out after all the openflow switches establish TCP/TLS session with the controllers, how much time it takes for a cluster of controllers to push their respective roles to a set of switches connected to them. This is calculated as the time difference between the first Hello packet transmitted by the controller cluster and the last role request message transmitted by the controller cluster, once the controllers are started. The test can be reiterated with different number of Switches.

4.2 Test Setup

4.2.1 Topology

A set of Switches connected to a controller cluster consisting of multiple controllers. The connection may be TCP or TLS. The Switches and the controllers are connected to the same LAN.

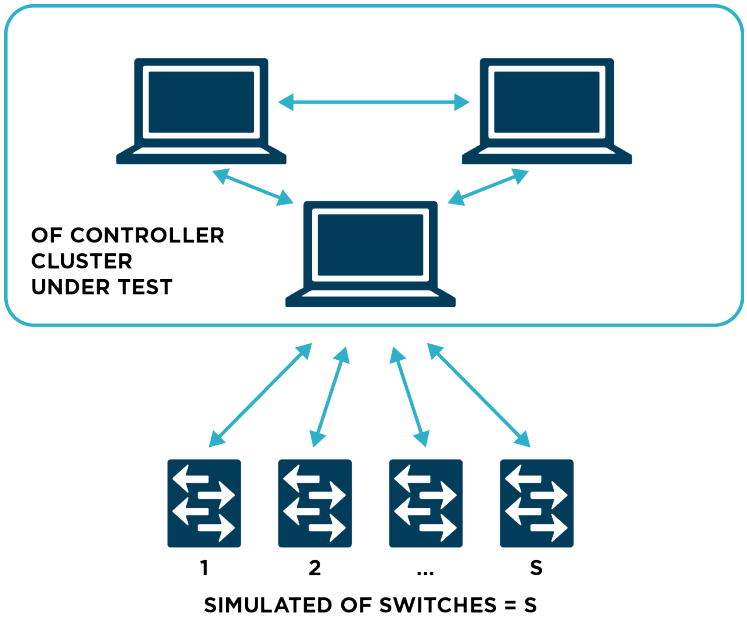


Figure 4.1: Test Topology for Role Intimation Time for a Controller Cluster

4.2.2 Prerequisites and Recommendations for the test

- The controller is directly connected (no additional IP hops) to all the switches (simulated by the test tool) to remove any error condition due to the other network activity and congestion.
- Use same switch simulator or real switches when testing with different controllers, so that switch side latencies remain a common factor for benchmarking different controllers from different vendors.

4.3 Test Configuration

4.3.1 Controller Configuration

The controller must be configured with the following configuration parameters to meet the objective of the test. Other configuration parameters must be kept at default. A few example iterations are defined in this document in the table below, but there can be multiple iterations with different combination of these parameters.

Table 4.1: Controller Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2	Iteration 3
Channel Type	TCP or TLS	TCP	TCP	TCP
Role Request	Must be able to send Role Request message once an OF channel is established with a switch.	Enabled	Enabled	Enabled
Topology Discovery	LLDP or any other protocol to discover topology	Enabled	Enabled	Disabled
Echo Request/Reply	Optional parameter. Depending on the frequency of transmission, it might affect the test outcome.	Disabled	Enabled with frequency n1/sec	Enabled with frequency n2/sec
Support multiple controller interaction	Must support interaction with other controller and elect new master when current master goes down	Enabled	Enabled	Enabled

4.3.2 Switch Configuration

On the test tool side, following switch parameters must be set before proceeding with the test case. Few combinations of configuration parameters for test iteration are given below, but there can be different combinations over which the test can be iterated.

Table 4.2: Switch Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2	Iteration 3
Channel Type	TCP or TLS	TCP	TLS	TCP
Multipart Reply	Should be able to reply to Multipart Request from controller.	Disabled	Enabled	Enabled

Echo Request/Reply	Optional parameter.	Disabled	Enabled with frequency n1/sec	Enabled with frequency n2/sec
Multiple Controller Support	Should be able to establish OF channel with multiple controllers	Enabled	Enabled	Enabled

4.4 Test Steps

1. Start packet capture on the switch side.
2. Start controller followed by the switch.
3. Wait till all the switches have learned the roles of all the controllers.
4. Stop capture.
5. Measure the time taken to push roles to all switches, refer to the Test Measurement section.
6. Re-iterate the test with different combinations.

4.5 Test Measurement

1. Note down the timestamp of the first Hello packet received by the switch emulator, which was sent by the controller cluster, let that be T1.
2. Note down the last role request received by switch emulator that was sent by the controller cluster. Let that be T2.
3. Time Taken to push role $= (T2 - T1)$ sec.

4.6 Test Results

Here is an example of presenting the data in tabular form:

Table 4.3: Test Results Format

Type of Channel	**LLDP**	**Hello**	**Periodic Echo Request**	**Multipart Request**	**Number of Switches**	**Time Taken to Push Role**
TCP	Disabled	Enabled	Enabled	Enabled	S1	T1 sec
					S2	T2 sec
					S3	T3 sec

4.7 Test Iterations: Combination of Variables

If the number of controllers are kept unchanged, the following variables might affect the outcome of the test and can be iterated over with different combinations.

Table 4.4: Test Iterations Format

Parameters	**Possible Options**
Channel Type	TCP or TLS
Number of Switches	10, 100, 1000 etc.
LLDP	Enabled/Disabled
Multipart Request	Enabled/Disabled
Echo Request frequency	1/sec, 100/sec etc. or Disabled

5 Test 5 Measure the Role Intimation Time for a Controller Cluster After Master Failover

5.1 Objective

After boot up a controller cluster does the master election for the first time. At that moment it also intimates the master role to the OpenFlow switch network for the first time. “Test 4: Measure the Role Intimation Time for a Controller Cluster At Startup” measures this “initial master election” role intimation time. In this test we measure the role intimation time when re-election of new master happens at some later point of time after boot up. The purpose of this test is to measure how quickly a newly elected master controller can intimate the master role to all the switches for which it has become the new master through re-election. This is calculated as the time difference between the first and the last role request message transmitted by a newly elected master after the previous master fails. The test can be reiterated with different number of Switches.

5.2 Test Setup

5.2.1 Topology

A set of Switches connected to a controller cluster consisting of multiple controllers. The connection may be TCP or TLS. The Switches and the controllers are connected to the same LAN.

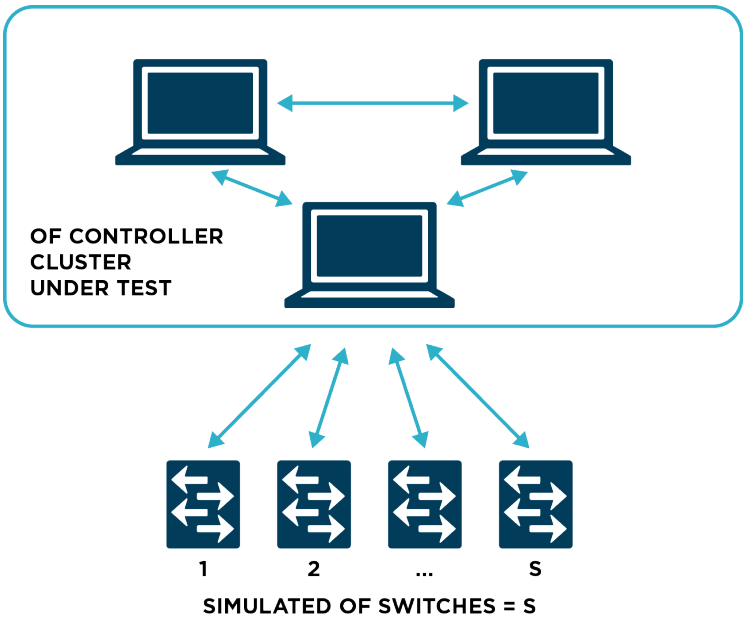


Figure 5.1: Test Topology for Role Intimation Time for a Controller Cluster

5.2.2 Prerequisites and Recommendations for the test

- The controller is directly connected (no additional IP hops) to all the switches (simulated by the test tool) to remove any error condition due to the other network activity and congestion.
- Use same switch simulator or real switches when testing with different controllers, so that switch side latencies remain a common factor for benchmarking different controllers from different vendors.

5.3 Test Configuration

5.3.1 Controller Configuration

The controller must be configured with the following configuration parameters to meet the objective of the test. Other configuration parameters must be kept at default. A few example iterations are defined in this document in the table below, but there can be multiple iterations with different combination of these parameters.

Table 5.1: Controller Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2	Iteration 3
Channel Type	TCP or TLS	TCP	TCP	TCP
Role Request	Must be able to send Role Request	Enabled	Enabled	Enabled

	message once an OF channel is established with a switch.			
Topology Discovery	LLDP or any other protocol to discover topology	Enabled	Enabled	Disabled
Echo Request/Reply	Optional parameter. Depending on the frequency of transmission, it might affect the test outcome.	Disabled	Enabled with frequency n1/sec	Enabled with frequency n2/sec
Support multiple controller interaction	Must support interaction with other controller and elect new master when current master goes down	Enabled	Enabled	Enabled

5.3.2 Switch Configuration

On the test tool side, following switch parameters must be set before proceeding with the test case. Few combinations of configuration parameters for test iteration are given below.

Parameter	Comments	Iteration 1	Iteration 2	Iteration 3
Channel Type	TCP or TLS	TCP	TLS	TCP
Multipart Reply	Should be able to reply to Multipart Request from controller.	Disabled	Enabled	Enabled
Echo Request/Reply	Optional parameter.	Disabled	Enabled with frequency n1/sec	Enabled with frequency n2/sec
Multiple Controller Support	Should be able to establish OF channel with multiple controllers	Enabled	Enabled	Enabled

Table 5.2: Switch Configuration Parameters

5.4 Test Steps

1. Start controller followed by the switch.
2. Wait till all the switches have learned the roles of all the controllers. At least one of the controller will take up the master role.
3. Start packet capture on the switch emulator side.
4. Bring down any one Master controller (master role may be distributed across the controllers in the cluster. If there are N switches, controller C1 may become master for N/2 switches and controller C2 may become master for other N/2 switches).
5. Wait till the newly elected master has sent role_request with master role.
6. Stop capture.
7. Measure the time taken to push roles to all switches concerned, refer to Test Measurement.

8. Re-iterate the test with different combinations and varying number of switches.

5.5 Test Measurement

1. Note down the timestamp of the first role_request received at switch emulator, sent by the newly elected master, let that be T1.
2. Note down the last role request received at switch emulator, let that be T2.
3. Time Taken to push role $= (T2 - T1)$ sec.

5.6 Test Results

Here is an example of presenting the data in tabular form:

Table 5.3: Test Results Format

Type of Channel	LLDP	Hello	Periodic Echo Request	Multipart Request	Number of Switches	Time Taken to Push Role
TCP	Disabled	Enabled	Enabled	Enabled	S1	T1 sec
					S2	T2 sec
					S3	T3 sec

5.7 Test Iterations: Combination of Variables

If the number of controllers are kept unchanged, the following variables might affect the outcome of the test and can be iterated over with different combinations.

Table 5.4: Test Iterations Format

Parameters	Possible Options
Channel Type	TCP or TLS
Number of Switches	10, 100, 1000 etc.
LLDP	Enabled/Disabled
Multipart Request	Enabled/Disabled
Echo Request frequency	1/sec, 100/sec etc. or Disabled

6 Test 6: Flow Setup Rate: Proactive Mode

6.1 Objective

The purpose of this test is to measure the rate by which an OF controller pushes a number of flows (configured statically in the controller) to a switch.

6.2 Test Setup

6.2.1 Topology

A single switch is connected to a controller, the connection may be TCP or TLS. There are two ports of the switch where two hosts are connected.

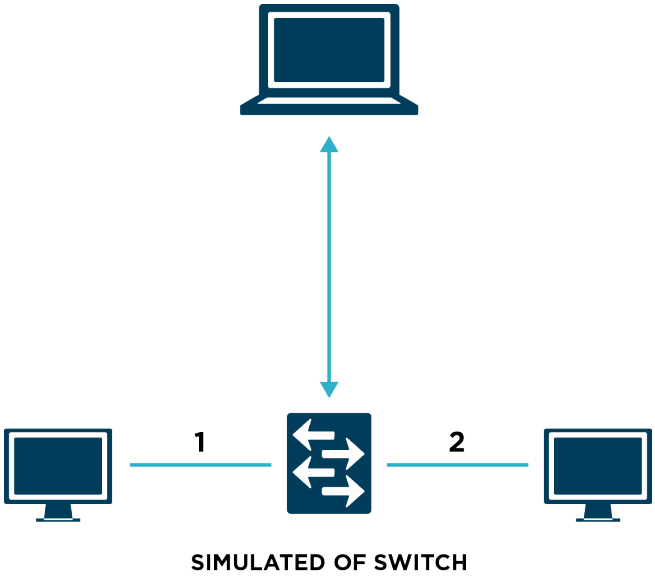


Figure 6.1: Test Topology for Flow Setup Rate: Proactive Mode

6.2.2 Prerequisites and Recommendations for the test

- The controller is directly connected (no additional IP hops) to the switch (simulated by the test tool) to remove any error condition due to the other network activity and congestion.
- Use same switch simulators or real switches, so that switch side latencies remain a common factor for benchmarking different controllers from different vendors.
- The test tool should be capable of capturing packet on the switch side.
- Controller should run application that can populate flows in the switches proactively upon administrator request.

6.3 Test Configuration

6.3.1 Controller Configuration

The controller must be configured with the following configuration parameters to meet the objective of the test. Other configuration parameters must be kept at default value. It is also assumed that the switches have single table configured, so all flows are pushed to the single table by the controller.

Table 6.1: Controller Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2	Iteration 3	Iteration 4
Channel Type	TCP or TLS	TCP	TLS	TCP	TLS
Echo Request/Reply	Optional parameter. Might affect test result depending upon frequency of transmission.	Enabled.	Enabled	Enabled.	Enabled
Barrier Request	Controller sends Barrier Request after every Flow Mod messages and waits for Barrier Reply from switch before sending the next Flow Mod message.	Enabled	Enabled	Enabled	Enabled
Flow Profile	There has to be large number of preconfigured flows in the controller	L2 Profile.	L2 Profile	L3 Profile	L3 Profile

6.3.2 Switch Configuration

Following Switch parameters need to be set before proceeding with the test case.

Table 6.2: Switch Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2	Iteration 3	Iteration 4
Channel Type	TCP or TLS as configured in controller	TCP	TCP	TLS	TLS
Echo Request/Reply	Optional parameter.	Enabled	Enabled	Enabled	Enabled
Barrier Reply	Should reply to Barrier Request received from controller.	Enabled	Enabled	Enabled	Enabled
Number of Switches	Total number of switches in the topology.	S1	S2	S3	S4

6.4 Test Steps

1. Configure F number of flows in the controller.
2. Start packet capture on the Switch.
3. Start the controller followed by the switches.
4. Wait for a pre-defined time period (flow push time) that the controller may take to push all the flows to the switch. This may be taken as a test input configurable by the user.
5. Verify on each switch that it has got F flows populated.
6. Stop Capture

7. For each switch, check the packet capture and find out the time between the first and the last flow mod message from controller.
8. Stop capture.

6.5 Test Measurement

1. Note down the time gap between the first Flow-mod and the last Flow-mod message sent out by the controller, let it be T sec.
2. There should be F number of flow-mod messages sent out by the controller.
3. Find out the rate as:

   Flow Push Rate = $(F)/T$ per sec.

6.6 Test Results

Here is some test output obtained:

Table 6.3: Results Format

Type of Channel	Echo Request	Multipart Request	Barrier Request	Flow Profile	Flow Push Rate (Flows/sec)
TCP	Disabled	Enabled	Disabled	L2	N1
TCP	Disabled	Enabled	Enabled	L2	N2

6.7 Test Iterations: Combination of Variables

The following variables might affect the outcome of the test and can be iterated over with different combinations of values.

Table 6.4: Iterations Format

Parameters	Possible Options
Channel Type	TCP or TLS
Type of Flow Profile	L2, IP, TCP
Periodic Echo	Enabled or Disabled (If enabled at what frequency)
Periodic Multipart Request	Enabled or Disabled
Barrier Request	Enabled/Disabled

7 Test 7: Flow Setup Rate: Reactive Mode

7.1 Objective

The purpose of this test is to measure the rate at which an OpenFlow controller sends flow_mod messages in response to large number of packet_in messages generated by switches connected to it. The controller under test needs to be configured to generate flow_mod in response to incoming packet_in. This test counts flow_mod messages and calculates the rate of transmission.

7.2 Test Setup

7.2.1 Topology

A single switch is connected to a single controller; the connection may be TCP or TLS (the test is also recommended to be iterated over multiple switches).

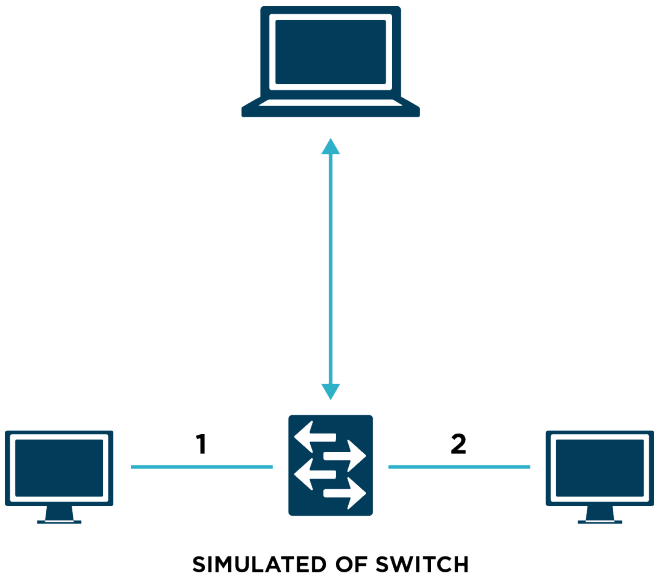


Figure 7.1: Test Topology for Flow Setup Rate: Reactive Mode

7.2.2 Prerequisites and Recommendations for the test

- The controller is directly connected (no additional IP hops) to the switch (simulated by the test tool) to remove any error condition due to the other network activity and congestion.
- Use same switch simulator or real switches when testing with different controllers, so that switch side latencies remain a common factor for benchmarking different controllers from different vendors.
- The test tool must be capable of capturing packet on the switch side.

7.3 Test Configuration

7.3.1 Controller Configuration

The controller must be configured with following configuration parameters to meet the objective of the test. Other configuration parameters must be kept at default. Few example iterations of the test are defined in the table below. There can be more iterations with different parameter combinations. It is assumed that the controller must run an application that sends flow_mod messages to switch in response to packet_in messages received. There should not be any flow aggregation done by the controller, that is, for each packet_in with unique L2/L3 header the controller sends a new flow_mod message.

Table 7.1: Controller Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2	Iteration 3	Iteration 4
Channel Type	TCP or TLS	TCP	TLS	TCP	TLS
Echo Request/Reply	Optional parameter.	Enabled.	Enabled	Disabled.	Disabled
Barrier Request	Controller sends Barrier Request after every Flow Mod messages and waits for Barrier Reply from switch before sending the next Flow Mod message.	Enabled	Enabled	Disabled	
Flow Profile	No predefined flow profile needed	NA	NA	NA	NA

7.3.2 Switch Configuration

Following Switch parameters must be set before proceeding with the test. Few combinations for iteration are illustrated below, but there can be different combinations over which the test can be iterated.

Table 7.2: Switch Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2	Iteration 3	Iteration 4
Channel Type	TCP or TLS as configured in controller	TCP	TCP	TLS	TLS
Echo Request/Reply	Optional parameter.	Enabled	Disabled	Enabled	Disabled
Barrier Reply	Should reply to Barrier Request received from controller.	Enabled	Disabled	Enabled	Disabled
Number of packet_in messages	Number of packet_in messages that the switch will send out once the OF channel is established with the controller	10k/switch	10k/switch	10k/switch	10k/switch
Packet_in Tx Rate	Rate at which Packet_in messages are sent out from the switch	1000/sec	1000/sec	1000/sec	1000/sec

7.4 Test Steps

1. Configure each switch such that it can send large number of packet_in messages (Consider, 10k per switch). These 10k packet_in messages must be divided in two sets, set 1 which should have in_port as 1, source mac X, and destination mac Y. Set 2 must have in_port as 2, source mac Y and destination mac X (that is, the reverse of set 1). Same configuration applies for IP addresses of L3 profiles, set 1 which should have in_port as 1, source ip x.x.x.x and destination ip y.y.y.y. Set 2 should have in_port as 2, source ip y.y.y.y and destination ip x.x.x.x (that is, the reverse of set 1). Sending bi-directional traffic may be required for some controllers who do not push flows if destination Host is not yet discovered and need to see packet from both the source and destination hosts.
2. Start controller.
3. Start the switches.
4. Once the OF channel/s is/are established, start packet capture on the switch simulator.
5. Wait till all the switches have sent out the packet_in messages configured. This can be verified either by packet_in Tx count (packet_in transmitted by switch) on each switch or counting the number of packet_in messages in the capture file.
6. Wait till the controller has replied to all the packet_in messages received. This can be verified either by flow_mod Rx count (flow_mod received by the switch) on each switch or counting the number of flow_mod messages received in the capture file. If there are N number of packet_in transmitted by the switch simulator, then there should be N flow_mod messages received by it.
7. Stop capture and check the packet capture to find out the time between the first and the last flow_mod message from controller.
8. Re-iterate the test with different combination of parameter values mentioned in 7.7.

7.5 Test Measurement

The following image depicts a time graph of the events happening during the test:

Note down the time gap between the first flow_mod and the last flow_mod message sent out by the controller. Let it be δ T sec.

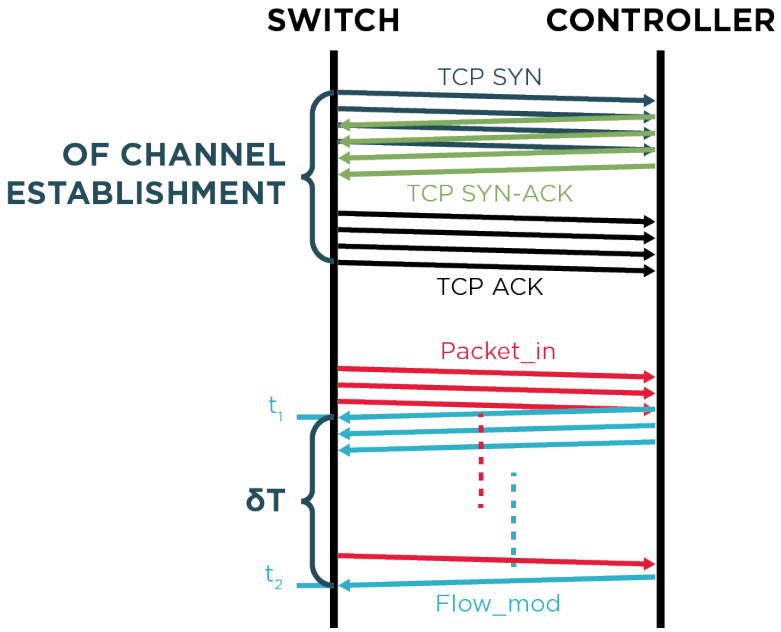


Figure 7.2: Time Diagram for Flow Setup Rate: Reactive Mode

1. There must be N number of flow_mod messages sent out by the controller in response to N packet_in messages received.
2. Find out the rate using:

   Flow Setup Rate = $(N)/\delta T)$ per sec.

7.6 Test Results

Here is some sample test output:

Table 7.3: Test Result Format

Type of Channel	Echo Request	Barrier Request	LLDP	Multipart Request	Header Type	Packet_in Tx Rate	Flow Setup Rate (flow/sec)
TCP	Enabled	Enabled	Enabled	Enabled	L2	1000/sec	N1
TCP	Disabled	Enabled	Disabled	Disabled	L2	1000/sec	N2

7.7 Test Iterations: Combination of Variables

If the number of switch is kept constant, the following variables might affect the outcome of the test and can be iterated over with different combinations. Iteration can be done with multiple number of switches to measure the factor affecting the flow push rate of the controller. The test can also be iterated with single table against multiple tables configured in the switch. If there are multiple tables in the switch and controller decides to push L2 flows in Table 0 and L3 flows in

Table 1, then flow push rate might affect for L3 flows. For L3 flow in single table, the controller pushes the flows in the table 0; whereas for multiple table it pushes the flows in multiple tables.

Table 8.5: Test Variables

Parameters	**Possible Options**
Channel Type	TCP or TLS
Type of Packet Header included in packet_in	L2, IP, TCP
Periodic Echo	Enabled or Disabled (If enabled at what frequency)
Periodic Multipart Request	Enabled or Disabled
Barrier Request	Enabled or Disabled
Packet_in message Tx rate	Different Tx rates such as 500/sec, 10k/sec etc.

8 Test 8: Flow Setup Rate with Auxiliary Channel: Reactive mode

8.1 Objective

The purpose of this test is to find out how the flow set up rate (no of flow/sec) is improved with the presence of multiple Auxiliary Channels between a controller and a Switch. "Test 7: Flow Setup Rate: Reactive Mode" measures the rate without any Auxiliary channel, the same test is repeated with the presence of Auxiliary channels, and compare whether there is any improvement or not in flow/sec.

8.2 Test Setup

8.2.1 Topology

A single switch is connected to a single controller having a main channel and N number of Auxiliary channels, the connection may be TCP or UDP.

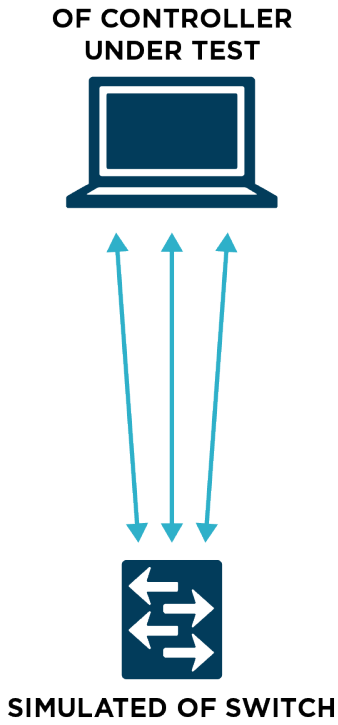


Figure 8.1: Test Topology for Flow Setup Rate with Auxiliary Channel

8.2.2 Prerequisites and Recommendations for the test

- The controller is directly connected (no additional IP hops) to the switch (simulated by the test tool) to remove any error condition due to the other network activity and congestion.

- Use same switch simulator or real switches when testing with different controllers, so that switch side latencies remain a common factor for benchmarking different controllers from different vendors.
- Both the controller and switch must support Auxiliary channels.
- Controller must run application which sends flow_mod messages to switches in response to packet_in messages received.
- It is recommended that the controller should send flow_mod messages on the same auxiliary connection over which the corresponding packet_in was received. But this is not a must. This test can also be done if flow_mod messages are sent out only over main channel.

8.3 Test Configuration

8.3.1 Controller Configuration

The controller must be configured with the following configuration parameters to meet the objective of the test. Other configuration parameters must be kept at default. A few example iterations are defined in this document in the table below, but there can be multiple iterations with different combination of these parameters. For this test, it is assumed that the controller will reply with flow_mod message for each packet_in that it receives and it is capable of using Auxiliary channels to transmit flow_mod messages.

Table 8.1: Controller Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2
Auxiliary Channel Type	TCP or UDP	TCP	UDP
Multipart Request	Periodic sending of Multipart Request may affect test result.	Enabled.	Enabled
Echo Request/Reply	Optional parameter.	Enabled.	Enabled
Barrier Request	Controller sends Barrier Request after every Flow Mod messages and waits for Barrier Reply from switch before sending the next Flow Mod message.	Enabled	Enabled
Number of Auxiliary channels	Number of Auxiliary channels between the Switch and controller	N1	N1

8.3.2 Switch Configuration

Following switch parameters must be set before proceeding with the test case. Few example set of values for iteration are illustrated below, but there can be different combinations over which the test can be iterated.

Table 8.2: Switch Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2
Auxiliary Channel Type	TCP or UDP	TCP	UDP
Multipart Reply	Should be able to reply to Stats Request from controller.	Enabled	Enabled
Echo Request/Reply	Optional parameter.	Enabled	Enabled
Barrier Reply	Should reply to Barrier Request received from Controller.	Enabled	Enabled
Number of Auxiliary Channels	Total number of Auxiliary Channels.	N1	N2
Number of packet_in messages	Number of packet_in messages that the switch will send out once the OF channel is established with the controller	10k/switch	10k/switch

8.4 Test Steps

1. Configure A1 number of Auxiliary channels between the switch and the controller.
2. Configure the switch such that it distributes *N* packet_in messages equally over the main+auxiliary channels with the same packet_in Tx rate configured on the switch. These *N* packet_in messages must be divided in two sets:
    a. Set 1, which should have in_port as 1, source mac X and destination mac Y
    b. Set 2 which should have in_port as 2, source mac Y and destination mac X (reverse of set 1).
9. Same configuration applies for IP addresses of L3 profiles; that is, set 1 should have in_port as 1, source ip x.x.x.x and destination ip y.y.y.y, and set 2 should have in_port as 2, source ip y.y.y.y and destination ip x.x.x.x (reverse of set 1) . Sending bi-directional traffic may be required for some controllers who do not push flows if destination Host is not yet discovered and need to see packet from both the source and destination hosts.
3. Start controller.
4. Start switch.
5. Start packet capture on the switch simulator.
6. Wait till the switch has sent out all the packet_in messages configured there. This can be verified by packet_in Tx statistics (packet_in messages transmitted by the switch) on the switch or by counting the number of packet_in messages in the capture file.
7. Wait till the controller has replied to all the packet_in messages received. This can be verified by looking at flow_mod Rx statistics (flow_mod messages received at the switch) on the switch or counting the number flow_mod messages in the capture file.
8. Stop capture.
9. Measure the flow setup rate (refer 8.5).
10. Repeat the test with A2 number of Auxiliary channels.

8.5 Test Measurement

1. Find out the flow_mod/sec over the main and auxiliary channel together from the packet capture as described below:
    a. Let the time difference between the first and last flow_mod received by the switch in packet capture is T2 sec.
    b. N is the total number of packet_in messages generated by the switch.
    c. Overall Flow per sec with auxiliary channel: $\boldsymbol{FPS(aux) = N/T2}$
2. Compare ***FPS(main)*** and ***FPS(aux)***, ***FPS(main)*** is the result of ***Test 6***.

8.6 Test Results

Here is an example of presenting the data in tabular form:

Table 8.3: Test Results Format

Type of Channel	Echo Request	Barrier Request	LLDP	Multipart Request	Header Type	Number of Auxiliary Chanel	Packet_in Tx Rate	Flow Setup Rate (flow/sec)
TCP	Enabled	Enabled	Enabled	Enabled	L2	0	1000/sec	N1
TCP	Enabled	Enabled	Enabled	Enabled	L2	A1	1000/sec	N2

8.7 Test Iterations: Combination of Variables

If the number of Auxiliary channels is kept constant, the following variables might affect the outcome of the test and can be iterated over with different combinations as mentioned in Test 6.

Table 8.4: Test Variables

Parameters	Possible Options
Channel Type	TCP or TLS
Type of Packet Header included in packet_in	L2, IP, TCP
Periodic Echo	Enabled or Disabled (If enabled at what frequency)
Periodic Multipart Request	Enabled or Disabled
Barrier Request	Enabled or Disabled
Packet_in message Tx rate	Different Tx rates such as 500/sec, 10k/sec, and so on.

9 Test 9: End-to-End Flow Setup Time

9.1 Objective

The purpose of this methodology is to find out the time taken by an OF controller to establish an end-to-end data path between a source host and a destination host connected across a multi hop network of OF switches. The end to end flow setup time is measured in terms of the time it takes to install all the flows, such that the data traffic can successfully reaches from a source host to a destination host through the switch network. The assumption made here is that the controller is a reactive controller and it has not installed any end-to-end flow proactively (for example proactively flooding is not set across the switches). The test measurement can be done with different types of data traffic like L2 MAC, IP, and TCP/UDP, and so on.

9.2 Test Setup

9.2.1 Topology

S number of switches connected to each other in a linear topology. Source host H_S is connected to a switch at one end of the topology and destination host H_D is connected to the other extreme end. There is a single OF controller connected to all the switches present in the given topology. It is recommended to have a linear topology such that the controller is stressed to the maximum for this topology. This is because, it has to push flows to all the switches in the topology, as opposed to any other topology; where it may not need to push flows to all switches to establish an end-to-end data path due to redundant paths.

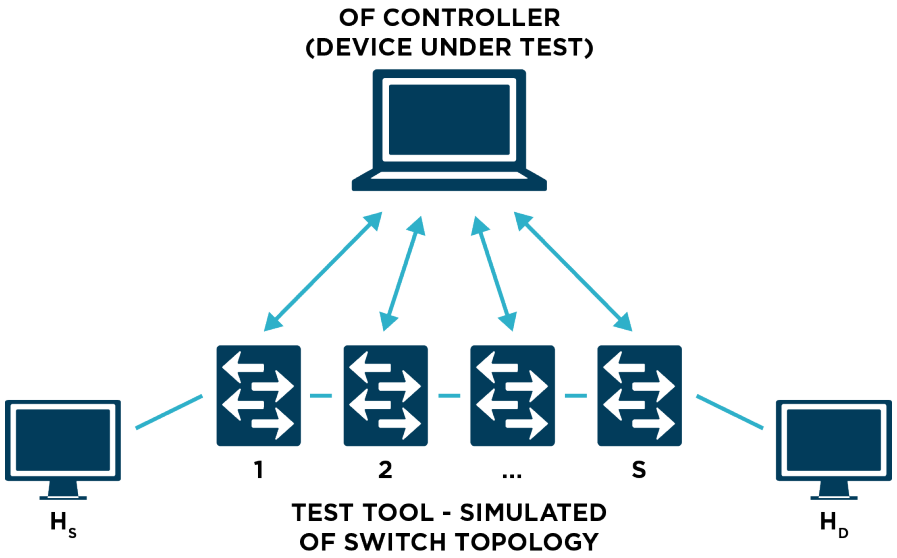


Figure 9.1: Test Topology for End-to-End Flow Setup Time

9.2.2 Prerequisites and Recommendations for the test

- The controller is directly connected (no additional IP hops) to all the switches (simulated by the test tool) to remove any error condition due to the other network activity and congestion.
- The controller must support control capture on the loopback interface to intercept the message exchange between the switches and itself.
- End hosts connected to OF switches must be capable of sending different types of data packet like L2 MAC, IP, TCP/UDP, and so on.
- The controller starts pushing the required flows to set up an end-to-end communication path, upon receiving the very first packet_in with a data packet.

9.3 Test configuration

9.3.1 Controller configuration

The controller must be configured with the following configuration parameters to meet the objective of this test.

Table 9.1: Controller Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2
Channel Type	TCP or TLS	TCP	TLS
Stats Request	Controller sends periodic Stats Request message to switch. Stats Reply from the switch acts as an indicator that channels established are in good health.	Enabled.	Enabled
Echo Request/Reply	Optional parameter.	Enabled.	Enabled
Barrier Request	Controller sends Barrier Request after every Flow Mod messages and waits for Barrier Reply from switch before sending the next Flow Mod message.	Enabled	Enabled
Topology Discovery	Should support switch topology discovery mechanism such as LLDP.	Enabled	Enabled
Packet_in/Packet_out	Should be able to process packet_in messages and send flow_mod messages.	Enabled	Enabled
Flow Profile	There are no preconfigured flows in the controller	No preconfigured Flow Profile.	No preconfigured Flow Profile.

9.3.2 Switch configuration

Following Switch parameters must be set before proceeding with the test case.

Table 9.2: Switch Configuration Parameters

Parameter	Comments	Iteration 1.1	Iteration 1.2	Iteration 2.1	Iteration 2.2
Channel Type	TCP or TLS as configured in controller	TCP	TCP	TLS	TLS
Stats Reply	Should be able to reply to Stats Request from controller.	Enabled	Disabled	Enabled	Disabled
Echo Request/Reply	Optional parameter.	Enabled	Disabled	Enabled	Disabled
Barrier Reply	Should reply to Barrier Request received from Controller.	Enabled	Disabled	Enabled	Disabled
Flow Miss Entry	The action corresponding to a flow miss entry should be able to generate packet_in with the received data packet and send to the controller.	Generate packet_in with data packet	Generate packet_in with data packet	Generate packet_in with data packet	Generate packet_in with data packet
Topology Discovery Support	Should support the topology discovery protocol that controller is using, such as LLDP.	Supported	Supported	Supported	Supported
Number of Switches	Total number of switches in the topology, which is kept constant	N	N	N	N

9.3.3 Host configuration

1. The host and switches are simulated by the test tool.
2. The host should have an IP configured on the interface connected to the switch.
3. The host should be capable of generating L2 MAC, IP, TCP/UDP traffic.

9.3.4 Test steps

1. Start switches and establish OF channel with the controller.
2. Wait for a pre-defined time till the controller has learned the topology. This wait time may be taken as a user input.
3. Start capture on the test tool.
4. From source host H_S, send an L2 traffic with destination MAC of host H_D and send another L2 data packet from host H_D with destination MAC of H_S.
5. Wait till the controller has sent out all the flow_mod messages required to transmit both the packets to their corresponding destination. This can be verified by checking

in flow_mod messages that has flow match criteria as the destination MAC address either of H_S or H_D.

6. Stop capture.

9.3.5 Test measurement

1. Note down the timestamp of the first packet_in sent out by switch-1 to controller with L2 header having destination MAC of H_D. Let that be T_1.
2. Note down the timestamp of the last flow_mod message that has flow match criteria as the destination MAC address either of H_S or H_D, let it be T_2.
3. End-to-End flow establishment time = $(T_2 - T_1)$ sec

9.4 Test Iterations: Combination of Variables

If the number of Switches (N) are kept constant, the following variables might affect the outcome of the test and can be iterated over with different combinations.

Table 9.3: Test Variables

Parameters	**Possible Options**
Channel Type	TCP or TLS
Type of Data Traffic	L2, IP, TCP
Periodic Echo	Enabled or Disabled (If enabled at what frequency)
Periodic Stats Request	Enabled or Disabled

10 Test 10: Controller Message Processing Latency

10.1 Objective

The purpose of this test is to find out the time taken by a controller to respond to different types of messages (For example, Packet_in, Echo Request) received from the switch. The test is done with single switch and then with multiple switches, to compare how it fares with minimum load and with high load.

10.2 Test Setup

10.2.1 Topology

S number of switches connected to a controller.

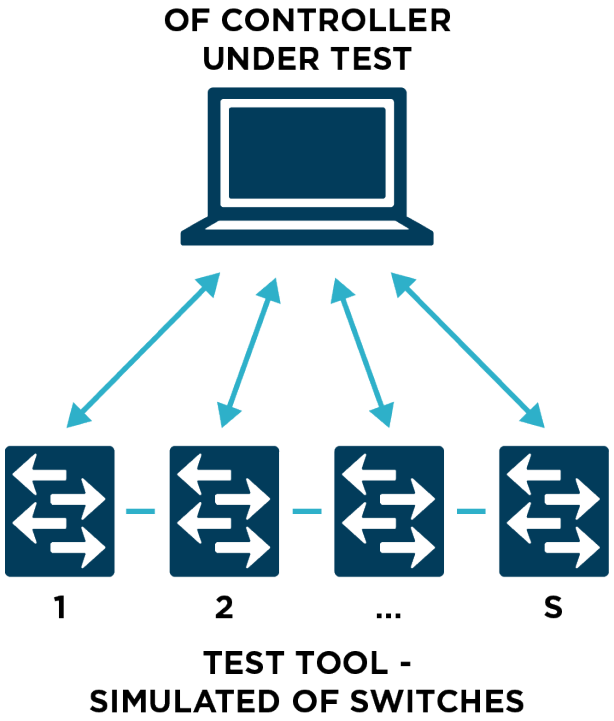


Figure 10.1: Test Topology for Controller Message Prcoessing Latency

10.2.2 Prerequisites and Recommendations for the test

- The controller is directly connected (no additional IP hops) to all the switches (simulated by the test tool) to remove any error condition due to the other network activity and congestion.
- Use same switch simulator or real switches when testing with different controllers, so that switch side latencies remain a common factor for benchmarking different controllers from different vendors.

- Controller should run an application that sends packet_out messages in response to packet_in message with ARP Request header.

10.3 Test Configuration

10.3.1 Controller Configuration

The controller must be configured with following configuration parameters to meet the objective of the test. Other configuration parameters must be kept at default. A few example iterations are defined in this document in the table below, but there can be multiple iterations with different combination of these parameters.

Table 10.1: Controller Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2	Iteration 3
Channel Type	TCP or TLS	TCP	TCP	TCP
Stats Request	Controller sends periodic Stats Request message to switch. Stats Reply from the switch acts as an indicator that channels established are in good health.	Disabled.	Enabled	Enabled
Echo Request/Reply	Optional parameter.	Disabled	Enabled with frequency n1/sec	Enabled with frequency n2/sec

10.3.2 Switch Configuration

On the test tool side, following switch parameters must be set before proceeding with the test case. Few example set of values for iteration are illustrated below, but there can be different combinations over which the test can be iterated.

Table 10.2: Switch Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2	Iterations 3
Channel Type	TCP or TLS	TCP	TCP	TCP
Stats Request	Controller sends periodic Stats Request message to switch. Stats Reply from the switch acts as an indicator that channels established are in good health.	Disabled.	Enabled	Enabled
Echo Request/Reply	Optional parameter.	Disabled	Enabled with frequency n1/sec	Enabled with frequency n2/sec

10.4 Test Steps

1. Start the controller.
2. Start packet capture on the test-tool port.
3. Start only a single switch.

4. Once the channel is established, send preconfigured n number of packet_in messages with ARP Request header.
5. Wait 5 min and then stop capture.
6. Measure the Latency as mentioned in the section 10.5.
7. Re-iterate the test with multiple switches, with all the switches are sending packet_in messages.
8. Re-iterate step 4 to step 7 with echo request message.

10.5 Test Measurement

1. From the control capture, record the timestamp of the first packet_in transmitted and the first packet_out message received by the switch. Find out the difference., Let it be T1 where;

$$T1 = TpktOut(1) - TpktIn(1)$$

2. Iterate few more time and let the measured times differences (as mentioned in the previous step) are, T2, T3 …. Tn.
3. Measure the Average Latency as:

$$\text{Average Latency} = (\sum_{k=1}^{n} Tk)/n$$

4. While testing with echo request message, measure the timestamp of echo request transmitted and echo reply received by the switch.

10.6 Test Results

Here is an example of presenting the data in tabular form:

Table 10.3: Test Results Format

Type of Channel	Message Type	Number of Switches	Message Processing Latency (Average)
TCP	Packet_in/Packet_out	1	T1 sec
		S	T2 sec
	Echo Request/Reply	1	T3 sec
		S	T4 sec

10.7 Test Iterations: Combination of Variables

Following variables might affect the outcome of the test and can be iterated over with different combinations.

Table 10.4: Test Iterations Format

Parameters	**Possible Options**
Channel Type	TCP or TLS
Message Tx Frequency	Frequency f1/sec, f2/sec
Number of OpenFlow Switches	

11 Test 11: Datapath Failure Recovery Time

11.1 Objective

The purpose of this methodology is to find out the time taken by a controller to setup an alternate path on the event of link failure in the existing path. This is measured in terms of the time elapsed between the port_status message sent out by the switch with link down information and the last flow_mod message received at the switch. From the beginning of the test there should be redundant paths between the source and the destination nodes. Initially, a flow path is setup by the controller and data traffic flows through the initial best path. Then a link on the initial best path is brought down, such that an alternate path can be established between the source and destination. The result of this depends heavily over the size of the topology, type of the topology, number of redundant paths, and the link brought down. Some predefined set of topologies are suggested over which the test can be iterated to find out the Datapath Failure Recover Time in different topologies.

11.2 Test Setup

11.2.1 Topology

S number of switches connected to each other in different types of topologies. One host H_S is connected to a switch at one end of the topology and another host H_D connected to the other extreme end. There is a single OF controller connected to all the switches present in the given topology.

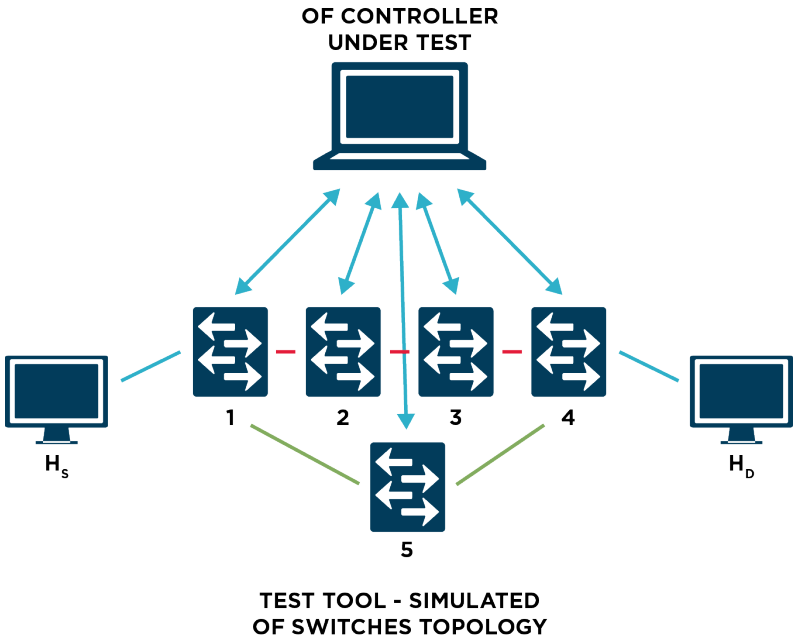


Figure 11.1: Test Topology for Datapath Failure Recovery Time

11.2.2 Prerequisites and Recommendations for the test

- The controller is directly connected (no additional IP hops) to all the switches (simulated by the test tool) to remove any error condition due to the other network activity and congestion.
- Use the same switch simulator or real switches when testing with different controllers, so that switch side latencies remain a common factor for benchmarking different controllers from different vendors.
- End Hosts connected to OF switches are capable of sending different types of data packet like L2 MAC, IP, TCP/UDP, and so on.

11.3 Test Configuration

11.3.1 Controller Configuration

The controller must be configured with the following configuration parameters to meet the objective of the test. Other configuration parameters must be kept at default.

Table 11.1: Controller Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2
Channel Type	TCP or TLS	TCP	TLS
Stats Request	Controller sends periodic Stats Request message to switch. Stats Reply from the switch acts as an indicator that channels established are in good health.	Enabled.	Enabled
Echo Request/Reply	Optional parameter.	Enabled.	Enabled
Barrier Request	Controller sends Barrier Request after every Flow Mod messages and waits for Barrier Reply from switch before sending the next Flow Mod message.	Enabled	Enabled
Topology Discovery	Should support switch topology discovery mechanism such as LLDP.	Enabled	Enabled
Packet_in/Packet_out	Should be able to process packet_in messages and reply with either packet_out or flow_mod messages.	Enabled	Enabled

11.3.2 Switch Configuration

The following switch parameters must be set before proceeding with the test case.

Table 11.2: Switch Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2
Channel Type	TCP or TLS as configured in controller	TCP	TLS
Stats Reply	Should be able to reply to Stats Request from controller.	Enabled	Enabled
Echo Request/Reply	Optional parameter.	Enabled	Enabled
Barrier Reply	Should reply to Barrier Request received from the controller.	Enabled	Enabled
Flow Miss Entry	The action corresponding to a flow miss entry should be to generate packet_in with the received data packet and send to the controller.	Generate packet_in with data packet	Generate packet_in with data packet
Packet_out	Should be able to process packet_out and take the action as mentioned in the packet_out message.	Packet_out processing enabled.	Packet_out processing enabled.

11.3.3 Host Configuration

1. The Host should have an IP configured on the interface connected to the switch.
2. Must be capable of generating L2 MAC, IP, TCP/UDP traffic.

11.4 Test Steps

1. Start the controller.
2. Start switches and establish OF channel with the controller.
3. Wait for a pre-defined time period (topology discovery time) that the controller may take to learn the topology. This may be taken as a test input configurable by the end user.
4. Ping H_D from H_S and verify it is successful or not. If unsuccessful, then abort the test and restart with a higher value of topology discovery time.

Initially let's say the Green path is selected for sending the packets to destination.

5. Start packet capture.
6. Now bring down the green connection between Switch1 and Switch5.
7. The ping results in failure once the connection is brought down.
8. Wait for a successful ping, indicating an alternate path has been setup.
9. There may be multiple flow_mod messages sent out by the controller to multiple switches.
10. Record the timestamp of first flow_mod message and the last flow_mod message received by any switch.
11. Repeat the test over a different topology.

11.5 Test Measurement

The following image depicts a time graph of the events happening during the test:

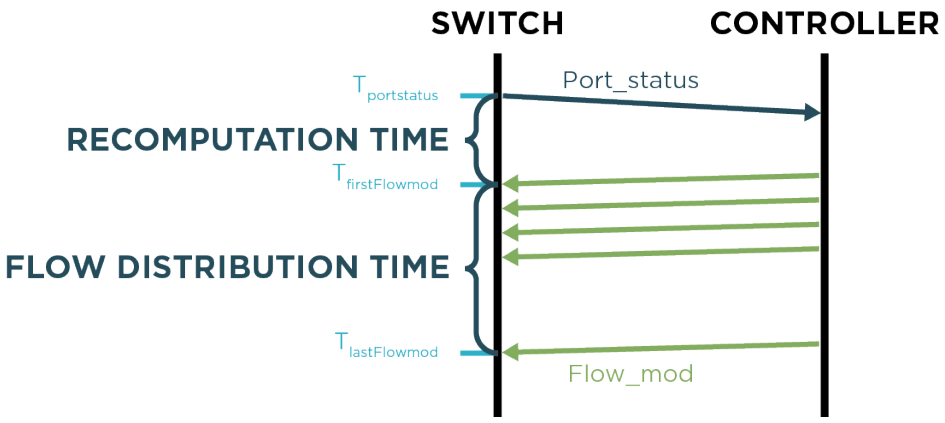


Figure 11.2: Time Diagram for Datapath Failure Recovery Time

1. Find out the time difference between first port_status message sent out by the switch and the first flow_mod message received by the switch after the port_status was transmitted.
2. This is actually the time taken by the controller to re-compute an end-to-end path between Hs and H_D. Let it be named as Path Re-computation Time and is calculated as below:

   $T_{recompute} = (TfirstFlowMod - TportStatus)$

3. Next find out the time difference between first flow_mod and last flow_mod message received by the switch and term it as Flow Distribution Time. This must be calculated as follows:

   $T_{flowdistribute} = (TlastFlowMod - TfirstFlowMod)$

4. Failure Recovery Time = Path Re-computation Time + Flow Distribution Time

11.6 Test Results

Here is an example of presenting the data in tabular form:

Table 11.3: Test Results Format

Number of Switches	Type of Topology	Failure Recovery Time
5	Ring	T1 sec
5	Fully Meshed	T2 sec
5	Leaf and Spine	T3 sec

11.7 Test Iterations: Combination of Variables

If the number of switches are kept unchanged, the following variables might affect the outcome of the test and can be iterated over with different combinations.

Table 11.4: Test Iterations Format

Parameters	**Possible Options**
Channel Type	TCP or TLS
Topology Type	Ring, Fully Meshed, Leaf and Spine (all having 5 nodes)
Periodic Echo	Enabled or Disabled (If enabled at what frequency)

12 Test 12: Datapath Switchover Time

12.1 Objective

The purpose of this methodology is to find out the time taken by a controller to setup a data path that is shorter/better than the existing one, due to addition of a new physical link in the topology. This is measured in terms of the time elapsed between the port status message sent out by the switches with link add information and the last flow_mod message received by the switches. These flow_mod messages are sent by the controller to create a better path which is now possible due to addition of new links to the topology.

Initially, there should be only one path between the source and the destination nodes. A flow path will be setup by the controller and data traffic flows through that. Then an additional link comes up as a result of which a shorter path becomes available for data transfer. This test measures how quickly controller identifies the shorter path and installs necessary flows in the switches such that the data traffic shifts to the newly established path. The result of this test depends heavily over the size of the topology, type of the topology, and the link added. Some predefined set of topologies are suggested over which the test can be iterated to find out the Datapath Switchover Time in different topologies.

12.2 Test Setup

12.2.1 Topology

S number of switches connects to each other in different types of topologies. One host H_S is connected to a switch at one end of the topology and another host H_D connected to the other extreme end. There is a single OF controller connected to all the switches present in the given topology.

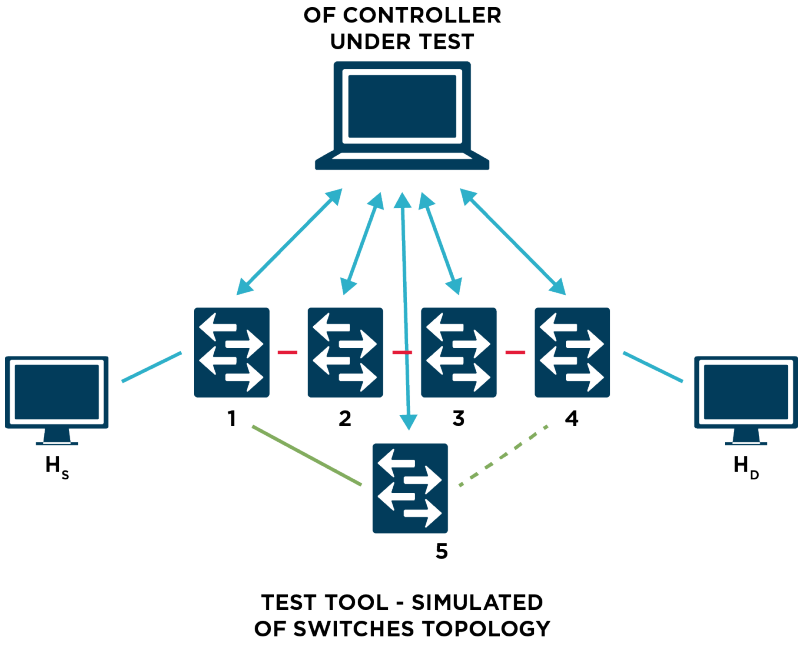


Figure 12.1: Test Topology for Datapath Switchover Time

12.2.2 Prerequisites and Recommendations for the test

- The controller is directly connected (no additional IP hops) to all switches (simulated by the test tool) to remove any error condition due to the other network activity and congestion.
- Use same switch simulator or real switches when testing with different controllers, so that switch side latencies remain a common factor for benchmarking different controllers from different vendors.
- End Hosts connected to OF switches are capable of sending different types of data packet like L2 MAC, IP, TCP/UDP, and so on.

12.3 Test Configuration

12.3.1 Controller Configuration

The controller must be configured with following configuration parameters to meet the objective of the test. Other configuration parameters should be kept at default.

Table 12.1: Controller Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2
Channel Type	TCP or TLS	TCP	TLS
Stats Request	Controller sends periodic Stats Request message to switch. Stats Reply from the switch acts as an	Enabled.	Enabled

	indicator that channels established are in good health.		
Echo Request/Reply	Optional parameter.	Enabled.	Enabled
Barrier Request	Controller sends Barrier Request after every Flow Mod messages and waits for Barrier Reply from switch before sending the next Flow Mod message.	Enabled	Enabled
Topology Discovery	Should support switch topology discovery mechanism such as LLDP.	Enabled	Enabled
Packet_in/Packet_out	Should be able to process packet_in messages and reply with either packet_out or flow_mod messages.	Enabled	Enabled

12.3.2 Switch Configuration

The following switch parameters must be set before proceeding with the test case.

Table 12.2: Switch Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2
Channel Type	TCP or TLS as configured in the controller	TCP	TLS
Stats Reply	Should be able to reply to Stats Request from the controller.	Enabled	Enabled
Echo Request/Reply	Optional parameter.	Enabled	Enabled
Barrier Reply	Should reply to Barrier Request received from the controller.	Enabled	Enabled
Flow Miss Entry	The action corresponding to a flow miss entry should be to generate packet_in with the received data packet and send to the controller.	Generate packet_in with data packet	Generate packet_in with data packet
Packet_out	Should be able to process packet_out and take the action as mentioned in the packet_out message.	Packet_out processing enabled.	Packet_out processing enabled.

12.3.3 Host Configuration

- The Host must have an IP configured on the interface connected to the switch.
- It must be capable of generating L2 MAC, IP, TCP/UDP traffic.

12.4 Test Steps – talk about ping

1. Start the controller.
2. Start switches and establish OF channel with the controller.
3. Wait for a pre-defined time period (Topology Discovery Time) that the controller may take to learn the topology. This may be taken as a test input configurable by the end user.
4. Ping H_D from H_S and verify if it is successful or not. If unsuccessful, then abort the test and restart with a higher value of Topology Discovery Time.

Initially let's say the Red path is selected for sending the packets to destination.

5. Start packet capture.
6. Now bring up a physical link between Swtic5 and Switch4 (dotted green line).
7. Both Switch4 and Switch5 will send Port Status message for the port addition event.
8. There may be multiple flow_mod messages sent out by the controller to multiple switches as a result of the new link addition.
9. Verify that the ping traffic now moves through the new data path.
10. Record the timestamp of first flow_mod message and the last flow_mod message received by any switch.
11. Repeat the test over a different topology.

12.5 Test Measurement

The following image depicts a time graph of the events happening during the test:

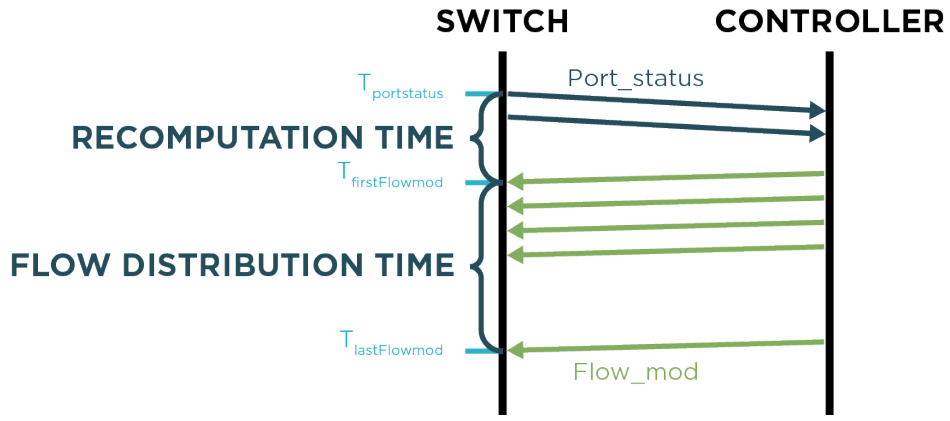


Figure 12.2: Time Diagram for Datapath Switchover Time

1. Find out the time difference between first port_status message sent out by the switch and the first flow_mod message received by the switch after the port_status was transmitted.
2. This is actually time taken by the controller to re-compute an end-to-end path between Hs and H_D. Let it be named as Path Re-computation Time and is calculated as below:

   $T_{recompute} = (TfirstFlowMod - TportStatus)$

3. Next, find out the time difference between first flow_mod and last flow_mod message received by the switch and term it as Flow Distribution Time. This should be calculated as follows:

   $T_{flowdistribute} = (TlastFlowMod - TfirstFlowMod)$

4. Datapath Switchover Time = Path Re-computation Time + Flow Distribution Time

12.6 Test Results

Here is an example of presenting the data in tabular form:

Table 12.3: Test Results Format

Number of Switches	Type of Topology	No Of Connecting Links	Failure Recovery Time
5	Ring	5	T1 sec
5	Fully Meshed	20	T2 sec
5	Leaf and Spine	10	T3 sec

12.7 Test Iterations: Combination of Variables

If the number of switches are kept unchanged, the following variables might affect the outcome of the test and can be iterated over with different combinations.

Table 12.4: Test Iterations Format

Parameters	Possible Options
Channel Type	TCP or TLS
Topology Type	Ring, Fully Meshed, Leaf and Spine (all having 5 nodes)
Periodic Echo	Enabled or Disabled (If enabled at what frequency)

13 Test 13: Data Path Link Connectivity Failure Recovery Time (Link Failure detected by LLDP and not by port status)

13.1 Objective

A controller discovers network topology through LLDP based topology discovery mechanism. LLDP based topology discovery runs periodically and a controller is supposed to detect any change in topology through the responses to LLDP flooding or through the lack of responses to LLDP flooding. This test measures when a controller detects any change in the previously discovered topology, then how fast the controller is able to take the corrective action. This topology change is detected by using LLDP and not through port status message.

At the beginning of the network boot up process, the controller has already discovered the currently active network topology. Then onwards, through periodic topology discovery mechanism the controller may discover additional changes on top of the topology already known by it. The changes in topology could be addition or deletion of new nodes into the topology. The changes could also be addition or loss of connectivity between two previously discovered nodes.

This test focuses on the situation where the controller detects (through LLDP and not through port status) any change in link connectivity between two previously discovered nodes and upon detection how fast the controller takes corrective action. If the initial best path was going through this link, then on detecting link connectivity issues, the controller should switch the data traffic through alternate path.

From the beginning of the test there should be redundant paths between the source and the destination nodes. Initially, a flow path is setup by the controller and data traffic flows through the initial best path (the green path traversing through switch 1-5-4 in the Figure 13.1: Test Topology for Datapath Failure Recovery Time below). Then the following actions can be taken to simulate loss of connectivity between the first two switches of the data path:

> Bidirectional loss of connectivity: Configure the simulated switch 1 such that it stops forwarding LLDP message to simulated switch 5. Also configure simulated switch 5 such that it stops forwarding LLDP message to simulated switch 1. As a result, controller will detect loss of connectivity in both directions between switch 1 and switch 5.

Once the controller detects loss of connectivity, it starts installing new flows such that the traffic shifts to the alternate path (1-2-3-4 in the diagram below). The result of this test depends heavily over the size of the topology, type of the topology, number of redundant paths, and the switches that stops processing packet_out message with LLDP. Some predefined set of topologies are suggested over which the test can be iterated to find out the Datapath Failure Recover Time in different topologies.

13.2 Test Setup

13.2.1 Topology

S number of switches connected to each other in different types of topologies. One host H_S is connected to a switch at one end of the topology and another host H_D connected to the other extreme end. There is a single OF controller connected to all the switches present in the given topology.

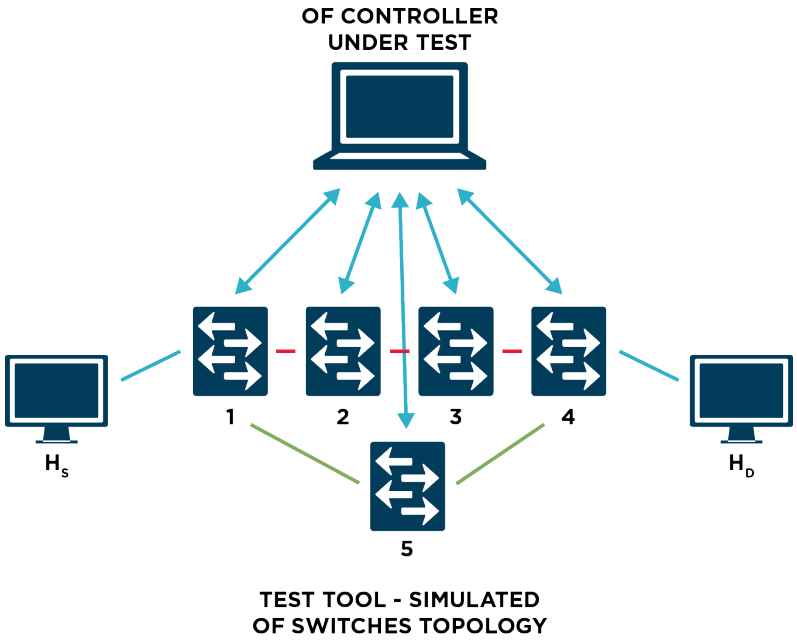


Figure 13.1: Test Topology for Datapath Failure Recovery Time

13.2.2 Prerequisites and Recommendations for the test

- The controller is directly connected (no additional IP hops) to all the switches (simulated by the test tool) to remove any error condition due to the other network activity and congestion.
- Use the same switch simulator or real switches when testing with different controllers, so that switch side latencies remain a common factor for benchmarking different controllers from different vendors.
- Switches capable of suppressing packet_out(LLDP) message on demand.
- End Hosts connected to OF switches are capable of sending different types of data packet like L2 MAC, IP, TCP/UDP, and so on.
- Controller uses LLDP for topology discovery and periodicity of the discovery process should be configurable.

13.3 Test Configuration

13.3.1 Controller Configuration

The controller must be configured with the following configuration parameters to meet the objective of the test. Other configuration parameters must be kept at default.

Table 13.1: Controller Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2
Channel Type	TCP or TLS	TCP	TLS
Echo Request/Reply	Optional parameter.	Enabled.	Enabled
Topology Discovery	Should support switch topology discovery mechanism such as LLDP.	Enabled	Enabled
SPF capable	Should be capable of SPF calculation and install flows accordingly.	Enabled	Enabled

13.3.2 Switch Configuration

The following switch parameters must be set before proceeding with the test case.

Table 13.2: Switch Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2
Channel Type	TCP or TLS as configured in controller	TCP	TLS
Echo Request/Reply	Optional parameter.	Enabled	Enabled
Flow Miss Entry	The action corresponding to a flow miss entry should be to generate packet_in with the received data packet and send to the controller.	Generate packet_in with data packet	Generate packet_in with data packet
Packet_out	Should be able to process packet_out and take the action as mentioned in the packet_out message.	Packet_out processing enabled.	Packet_out processing enabled.

13.3.3 Host Configuration

1. The Host should have an IP configured on the interface connected to the switch.
2. Must be capable of generating L2 MAC, IP, TCP/UDP traffic.

13.4 Test Steps

1. Configure the topology discovery periodicity at controller to 30 sec. (this is not a must, but for ease of measurement this is recommended, otherwise there will be too much LLDP packet exchange going on and it will be tough to figure out when a particular iteration starts and finishes.) Can be taken as a test input.
2. Start the controller.
3. Start switches and establish OF channel with the controller.
4. Wait for a pre-defined time period (topology discovery time) that the controller may take to learn the topology. This may be taken as a test input configurable by the end user.
5. Ping H_D from H_S and verify it is successful or not. If unsuccessful, then abort the test and restart with a higher value of topology discovery time.
6. Initially let's say the Green path (H_S-1-5-4-H_D) is selected for sending the packets to destination.
7. Start packet capture at the port where switches are simulated.
8. Wait once a particular topology discovery iteration finishes (can be verified by the packet_out Rx counter at the switch).
9. Configure switch 1 and switch 5 such that they don't forward LLDP messages to each other over the link connecting 1 and 5.
10. Wait for the next topology discovery iteration starts. Once started, controller is supposed to detect broken connection between switch 1 and switch 5 and install alternate data path.
11. Look for new flow_mod messages sent out by the controller to install new data path that traverses through 1-2-3-4.
12. Ping H_D from H_S and check the traffic traversed through 1-2-3-4.
13. There may be multiple flow_mod messages sent out by the controller to multiple switches.
14. Take measurement as mentioned in the following section.
15. Repeat the test over a different topology.

13.5 Test Measurement

The following image depicts a time graph of the events happening during the test:

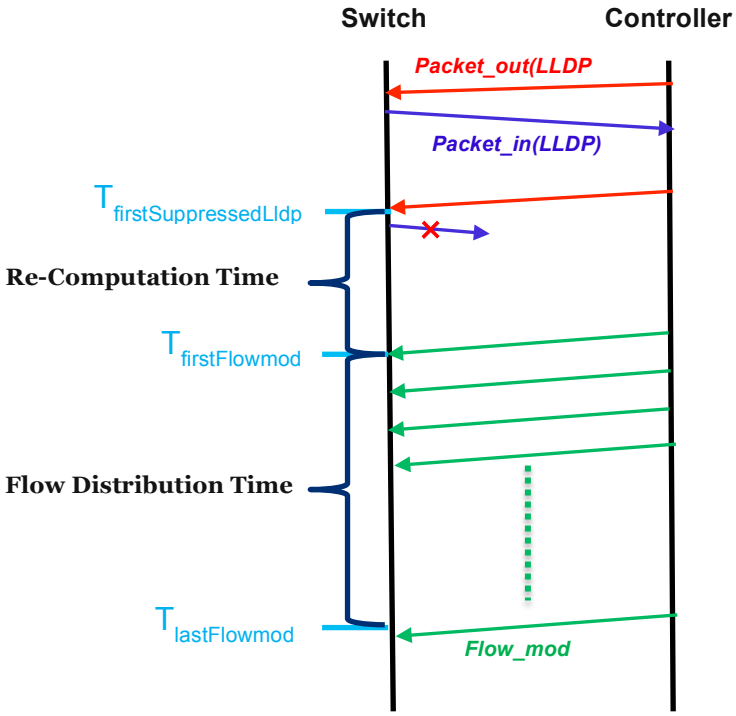


Figure 13.2: Time Diagram for Datapath Failure Recovery Time

1. Find out the first packet_out(LLDP) message received by switch 1 or switch 5 that was suppressed (whichever is received first), let the receive timestamp of the packet is $T_{firstSuppressedLldp}$. Whether switch 1 or switch 5 has suppressed the LLDP can be verified by taking the following steps:
    a. Lets say the port P1 of switch 1 is connected to port P2 of switch 5. Now switch 1 receives a packet_out message with a LLDP packet and action set to output with "out port" P1.
    b. As switch 1 is configured to suppress any LLDP message to be sent out to switch 5, switch 5 does not receive any LLDP and hence it will not send any packet_in message with LLDP message to the controller with "in port" set to P2.
    c. Similarly, switch 1 will not send any packet_in message with LLDP to the controller with "in port" set to P1.
2. Next find out the first flow_mod message received by the switch after $T_{firstSuppressedLldp}$. Let that be $T_{firstFlowMod}$
3. This is actually the time taken by the controller to re-compute an alternate end-to-end path between Hs and H_D. Let it be named as "Path Re-computation Time" and is calculated as below:

    $T_{recompute} = (TfirstFlowMod - TfirstSuppressedLldp)$

4. Next find out the time difference between first flow_mod and last flow_mod message received by the switch and term it as "Flow Distribution Time". This must be calculated as follows:
   $T_{flowdistribute} = (TlastFlowMod - TfirstFlowMod)$
5. Failure Recovery Time = Path Re-computation Time + Flow Distribution Time

13.6 Test Results

Here is an example of presenting the data in tabular form:

Table 13.3: Test Results Format

Number of Switches	**Type of Topology**	**Failure Recovery Time**
5	Ring	T1 sec
5	Fully Meshed	T2 sec
5	Leaf and Spine	T3 sec

13.7 Test Iterations: Combination of Variables

If the number of switches are kept unchanged, the following variables might affect the outcome of the test and can be iterated over with different combinations.

Table 13.4: Test Iterations Format

Parameters	**Possible Options**
Channel Type	TCP or TLS
Topology Type	Ring, Fully Meshed, Leaf and Spine (all having 5 nodes)
Periodic Echo	Enabled or Disabled (If enabled at what frequency)

14 Test 14: Datapath Switchover Time (Link addition detected by LLDP and not by port status message)

14.1 Objective

The purpose of this methodology is to find out the time taken by a controller to setup a data path that is shorter/better than the existing one, due to addition of a new link in the topology. Addition of new link is detected through periodic LLDP flooding. What this test will measure is, once addition of link to the topology is detected by the controller using LLDP then how fast the controller takes the corrective action to establish better end to end connectivity.

At the beginning of the test the controller will learn the active topology and the active connectivity among the nodes. Initially some of the nodes will be discovered. The LLDP packets exchanged on certain links will be suppressed by the test tool. Hench the controller will perceive these links to be nonexistent. As given in Figure 14.1: Test Topology for Datapath Failure Recovery Time initially the link between switch 1 and 5 will be nonexistent as the test tool will suppress the LLDP packets which will be flooded through link between 1 and 5. At this stage the controller will calculate 1-2-3-4 as the best path between H_S and H_D.

In the next topology discovery cycle the test tool allows the LLDP flooding to take place over the link connecting switch 1 and 5. This will make the controller to detect new connectivity that results in a better/shorter path than the existing one (1-2-3-4). The controller will start installing new flows such that the traffic shifts to the alternate path (1-5-4 in Figure 14.1: Test Topology for Datapath Failure Recovery Time).

The result of this test depends heavily over the size of the topology, type of the topology, number of redundant paths, and the switches that stop processing packet_out message with LLDP. Some predefined set of topologies are suggested over which the test can be iterated to find out the Datapath Switchover Time in different topologies.

14.2 Test Setup

14.2.1 Topology

S number of switches connected to each other in different types of topologies. One host H_S is connected to a switch at one end of the topology and another host H_D connected to the other extreme end. There is a single OF controller connected to all the switches present in the given topology.

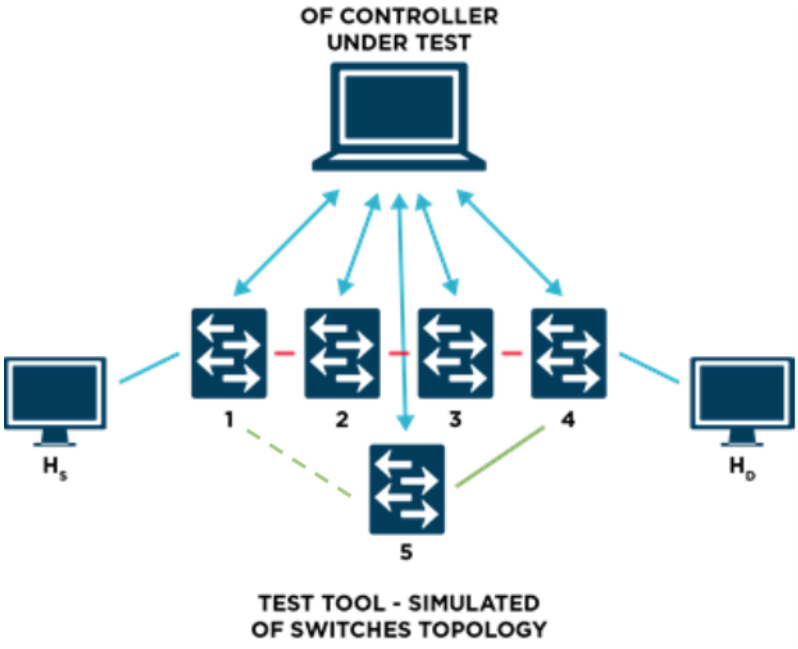


Figure 14.1: Test Topology for Datapath Failure Recovery Time

14.2.2 Prerequisites and Recommendations for the test

- The controller is directly connected (no additional IP hops) to all the switches (simulated by the test tool) to remove any error condition due to the other network activity and congestion.
- Use the same switch simulator or real switches when testing with different controllers, so that switch side latencies remain a common factor for benchmarking different controllers from different vendors.
- Switches capable of suppressing packet_out(LLDP) message and starts processing again on demand.
- End Hosts connected to OF switches are capable of sending different types of data packet like L2 MAC, IP, TCP/UDP, and so on.
- Controller uses LLDP for topology discovery and periodicity of the discovery process should be configurable.

14.3 Test Configuration

14.3.1 Controller Configuration

The controller must be configured with the following configuration parameters to meet the objective of the test. Other configuration parameters must be kept at default.

Table 144.1: Controller Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2
Channel Type	TCP or TLS	TCP	TLS
Echo Request/Reply	Optional parameter.	Enabled.	Enabled
Topology Discovery	Should support switch topology discovery mechanism such as LLDP.	Enabled	Enabled
SPF capable	Should be capable of SPF calculation and install flows accordingly.	Enabled	Enabled

14.3.2 Switch Configuration

The following switch parameters must be set before proceeding with the test case.

Table 144.2: Switch Configuration Parameters

Parameter	Comments	Iteration 1	Iteration 2
Channel Type	TCP or TLS as configured in controller	TCP	TLS
Echo Request/Reply	Optional parameter.	Enabled	Enabled
Flow Miss Entry	The action corresponding to a flow miss entry should be to generate packet_in with the received data packet and send to the controller.	Generate packet_in with data packet	Generate packet_in with data packet
Packet_out	Should be able to process packet_out and take the action as mentioned in the packet_out message.	Packet_out processing enabled.	Packet_out processing enabled.

14.3.3 Host Configuration

1. The Host should have an IP configured on the interface connected to the switch.
2. Must be capable of generating L2 MAC, IP, TCP/UDP traffic.

14.4 Test Steps

1. Configure the topology discovery periodicity at controller to 30 sec. (this is not a must, but for ease of measurement this is recommended, otherwise there will be too much

lldp packet exchange going on and it will be tough to figure out when a particular iteration starts and finishes.) Can be taken as a test input.

2. Start the controller.
3. Start switches and establish OF channel with the controller.
4.
5. Configure switch 1 and switch 5 such that they don't forward LLDP messages to each other.
6. Wait for a pre-defined time period (topology discovery time) that the controller may take to learn the topology. This may be taken as a test input configurable by the end user.
7. Ping H_D from H_S and verify it is successful or not. If unsuccessful, then abort the test and restart with a higher value of topology discovery time.
8. Initially the data traffic will traverse through the red path (H_S-1-2-3-4-H_D) since the dotted green path is not detected by controller as switch 1 and switch 5 are suppressing LLDP messages to each other.
9. Start packet capture at the port where switches are configured.
10. Now configure switch 1 and switch 5 to start allowing LLDP flooding among them.
11. Wait for next iteration of topology discovery to begin and look for switch 1 or switch 5 sending packet_in message with LLDP and "in port" set to the respective port connecting switch 1 and switch 5. As a result, controller will not detect the dotted green link between switch 1 and switch 5.
12. Look for new flow_mod messages sent out by the controller to switch to the shorter path available now (1-5-4).
13. Take measurement as mentioned in the following section.
14. Repeat the test over a different topology.

14.5 Test Measurement

The following image depicts a time graph of the events happening during the test:

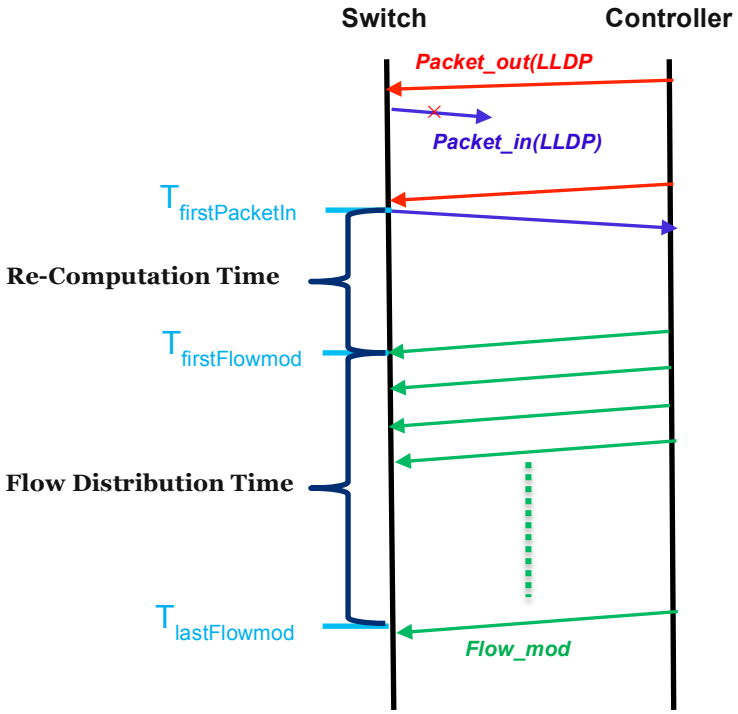


Figure 14.2: Time Diagram for Datapath Switchover Time

1. Find out the first packet_in(LLDP) message received by S1 or S5 that has "in port" set to the respective port connecting switch 1 and switch 5 (whichever is received earlier). Let the receive timestamp of the packet be $T_{firstPacketIn}$. Whether switch 1 or switch 5 has started allowing the LLDP messages amongst each other, can be verified by taking the following steps:
    a. Lets say the port P1 of switch 1 is connected to port P2 of switch 5. Now switch 1 receives a packet_out message with a LLDP packet and action set to output with "out port" P1.
    b. Switch 1 will forward the LLDP packet to switch 5 which will send a packet_in message with "in port" set to P5 with LLDP inside.
    c. Similarly, switch 5 will forward LLDP packet to switch 1 which will send a packet_in message with "in port" set to P1 with LLDP inside.
2. Next find out the first flow_mod message received by the switch after $T_{firstPacketIn}$. Let that be $T_{firstFlowMod.}$

3. This is actually the time taken by the controller to re-compute an alternate end-to-end path between Hs and H_D. Let it be named as "Path Re-computation Time" and is calculated as below:
   $T_{recompute} = (TfirstFlowMod - TfirstPacketIn)$
4. Next find out the time difference between first flow_mod and last flow_mod message received by the switch and term it as "Flow Distribution Time". This must be calculated as follows:
   $T_{flowdistribute} = (TlastFlowMod - TfirstFlowMod)$
5. Failure Recovery Time = Path Re-computation Time + Flow Distribution Time

14.6 Test Results

Here is an example of presenting the data in tabular form:

Table 144.3: Test Results Format

Number of Switches	Type of Topology	Switchover Time
5	Ring	T1 sec
5	Fully Meshed	T2 sec
5	Leaf and Spine	T3 sec

14.7 Test Iterations: Combination of Variables

If the number of switches are kept unchanged, the following variables might affect the outcome of the test and can be iterated over with different combinations.

Table 144.4: Test Iterations Format

Parameters	Possible Options
Channel Type	TCP or TLS
Topology Type	Ring, Fully Meshed, Leaf and Spine (all having 5 nodes)
Periodic Echo	Enabled or Disabled (If enabled at what frequency)

15 References

[1] TR_Requirements Analysis for Transport OpenFlow/SDN_v.1.0, Aug. 20, 2014.
[2] ITU-T Recommendation G.7714.1/Y.1705.1, “Protocol for automatic discovery in SDH and OTN networks”, 9/2010.
[3] OpenFlow Switch Specification 1.3.4 : https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.3.4.pdf

LIST OF CONTRIBUTORS

- Suvendu Mozumdar
- Kingshuk Mandal

