Performance Test White Paper for SDN Controller

Performance Test Methodologies and results for Open-source SDN Controller

Global SDN Certified Testing Center

December 11, 2015

1. INTRODUCTION

As Software Defined Network(SDN) being a new paradigm shifts in the network world, the separation of control plane and data plane makes the latter significantly simplified and lay off all the complex processing workload to the control plane. As the core element of control plane, SDN controller plays a vital role in the overall performance of SDN network. Since the deployment of SDN network kicks off, network users take the performance of SDN controllers into more and more serious considerations.

This white paper includes the detailed test methodologies and results for one open-source SDN controller regarding the performance particularly based on the OpenFlow 1.3.4 southbound interface protocol. Performance is the focus where the industry pays a lot of attention to since it's the vital influential factor for fulfilling the promise of SDN paradigm. The main purpose of this white paper is to firstly introduce performance test methodologies which could be replicated by network users and test labs to perform benchmark testing and secondly, try to provide quantitative analysis of one popular open-source controller under a particular software/hardware setup in lab environment.

The structure of this white paper is as follows: Section 2 talks about the test environment setup, software and hardware. Section 3 offers some general test recommendations for obtaining accurate and confident test results. Section 4 includes all the detailed information of performance test cases and results which have been implemented. Section 5 is an introduction to some extra performance test cases which are also valuable but cannot be implemented yet. We'd like to share the methodologies and seek for more collaboration. Section 6 is the conclusion and future plan. For each test case in Section 4, the white paper also compares the results of one single controller instance and three instances running in cluster mode.

All the tests and results statistics are implemented independently by Global SDN Certified Testing Center (SDNCTC).

2. TEST SETUP

2.1. Controller Under Test

An open-source SDN controller has been selected as the controller under test. It is one of the most popular SDN controllers currently and supports various functional features such as device management, topology management, path management, clustering, GUI and lots of build-in SDN applications. Its process capability of OpenFlow 1.3.4 messages is the point of this white paper. In this test we setup three identical virtual machines and each of them runs one identical instance of this controller.

2.2. Physical Server/Virtual Machine Configuration

- Physical Server Configuration: Dell PowerEdge R720
  CPU: Intel Xeon(R) CPU E5-2609 v2 @2.50Ghz 8 cores
  Memory: 16GB 1333Mhz
  OS: VMware ESXi 5.5 Update 2 Dell Customized, A00
- Virtual Machine Configuration:
  CPU: Intel Xeon(R) CPU E5-2609 v2 @2.50Ghz 4 cores
  Memory: 8 GB 1333Mhz
  OS: Ubuntu Server 14.04 TLS

2.3. Controller Operation Mode

In order to obtain a more comprehensive view of the performance of the controller, two different controller operation modes are defined:

- Standalone mode: One controller instance running in a virtual machine defined in Section 2.2.
- Cluster mode: Each virtual machine runs one controller instance. Three distributed controller instances running in different independent virtual machines are working distributive as one controller instance.

  Controller should be able to run in both standalone mode and cluster mode in order to verify the scalability and compare the performance differences.

2.4. Southbound/Northbound Protocol

The southbound protocol focused in this white paper is OpenFlow 1.3.4 only. Test cases are designed dedicated to this protocol. The northbound protocols are depending on controller's own implementation.

2.5. Test Tool

The test tools for implementing all the test cases are supported by commercial test tool vendors, specifically, Spirent and IXIA. Test cases are implemented by their dedicated OpenFlow SDN controller performance test tools. One particular test case is performed by one particular test tool.

3. TEST RECOMMENDATIONS

3.1. Isolated Recommendation

All the tests are recommended to be performed in lab environment within an isolated network in order to protect the accuracy of test results from any external interference.

3.2. Connectivity Recommendations

The controller under test is recommended to be connected directly with the emulated switches/test tools in order to avoid delays or failures introduced by the intermediate devices.

3.3. Repeatability Recommendations

All the performance tests are recommended to be iteratively implemented at least 10 times in order to get the confident average results.

3.4. Load Balance Verification Recommendations

When controller under test is running in cluster mode, the CPU and memory usage of each instance should be recorded in order to verify if there’s load balance mechanism existing for instances running in cluster mode when they are under process pressure.

4. PERFORMANCE TEST & RESULTS

4.1. Control Channel Capacity Test

PURPOSE

Benchmark the maximum number of OpenFlow control channels which could be established by a controller simultaneously.

PREREQUISITE

- The Controller under test should be able to run in both standalone mode and cluster mode.
- Both of the controller under test and the emulated switches should be able to maintain the aliveness of the control channels by using OpenFlow Echo request/reply messages.

RESULT EXPECTATION

- The maximum number of control channels established by a controller cluster should be no less than the number of the same controller running in standalone mode.

TOPOLOGY

In this test, certain number of independent switch nodes is setup and connected to the controller under test. There are no topology requirements for these nodes.

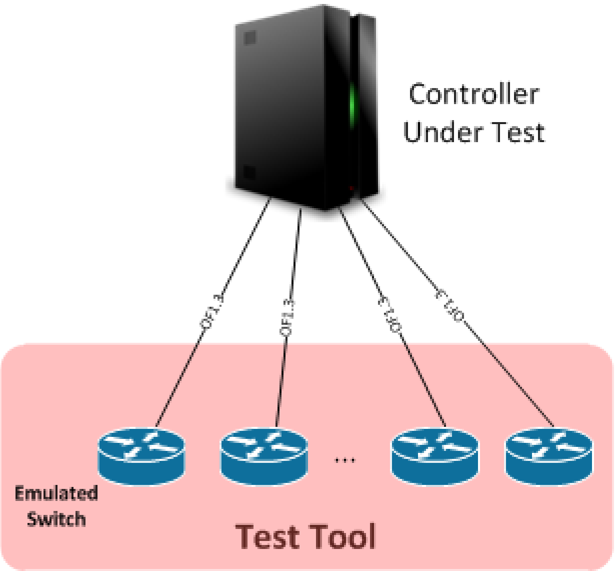


Fig 1.Topology for Control Channel Capacity test

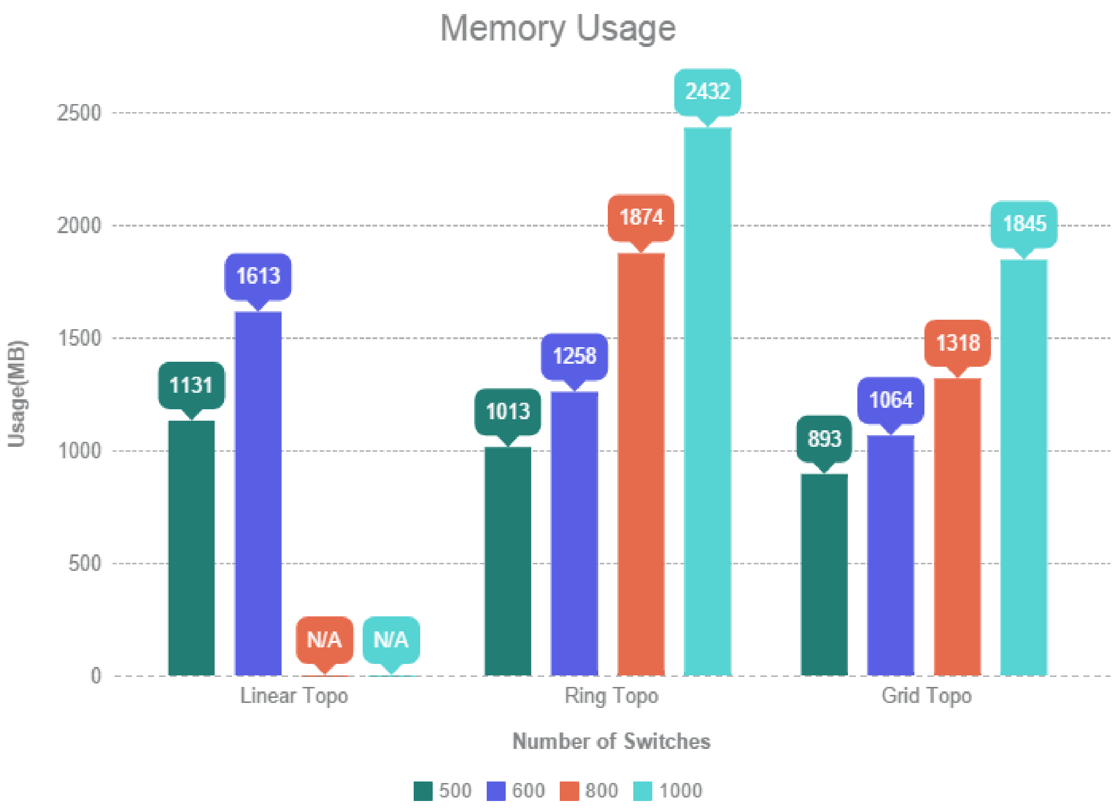
METHODOLOGY

1. Start one controller running in standalone mode
2. Start capture
3. Switches connected by different topologies start to establish connections to the controller
4. Increase the number of connected switches step by step until it reaches the maximum number Nm, record the average memory usage during this process
5. Wait for 5 minutes in order to ensure the established control channels are stable.
6. Iterate this test for another topology and controller instances running in cluster mode

RESULTS

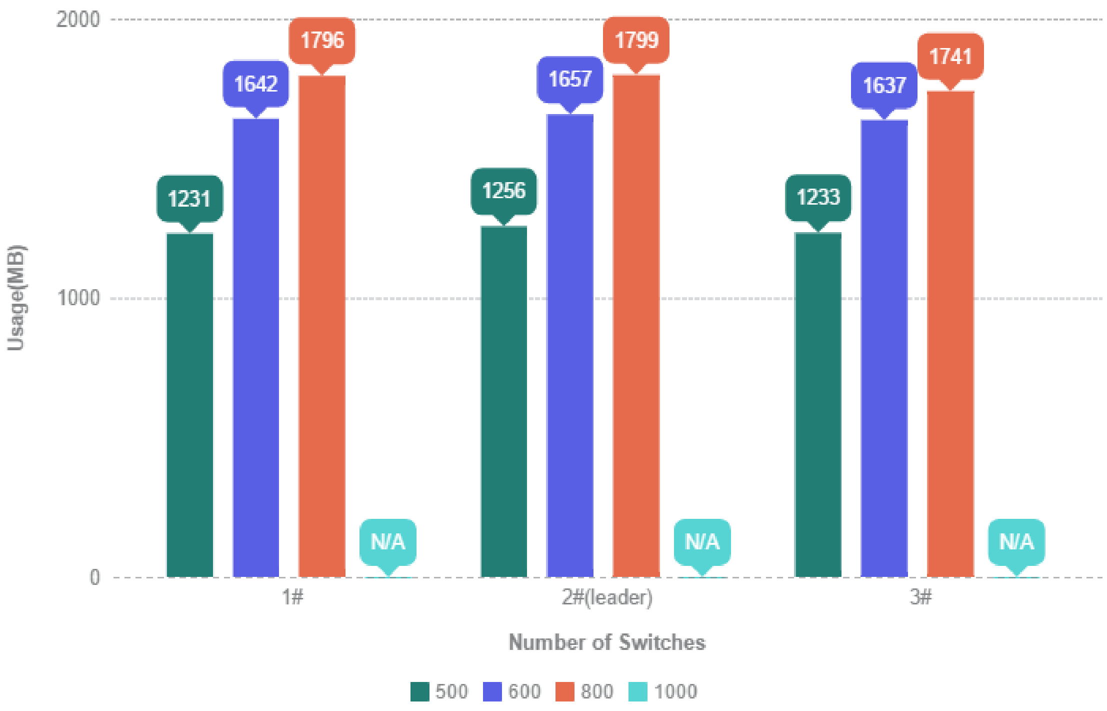
- Standalone Mode:

  Three kinds of topologies are implemented in this test case: linear topology, ring topology and grid topology. If the control channels are stable and all the switches are setup correctly by the controller, we will record the memory usage as a valid data. By default, the controller installs two flow entries for ARP packets and LLDP packets respectively to the switch after the establishment of the control channel. If there are switches do not receive these two Flow_add messages completely, we will consider the current number of switches is exceeds the capacity of the control channel and mark the usage as N/A. Regarding the ring topology and grid topology, when the number of switches increases to 1200, both of their results are N/A. The detailed average memory usage of this test case is shown below:

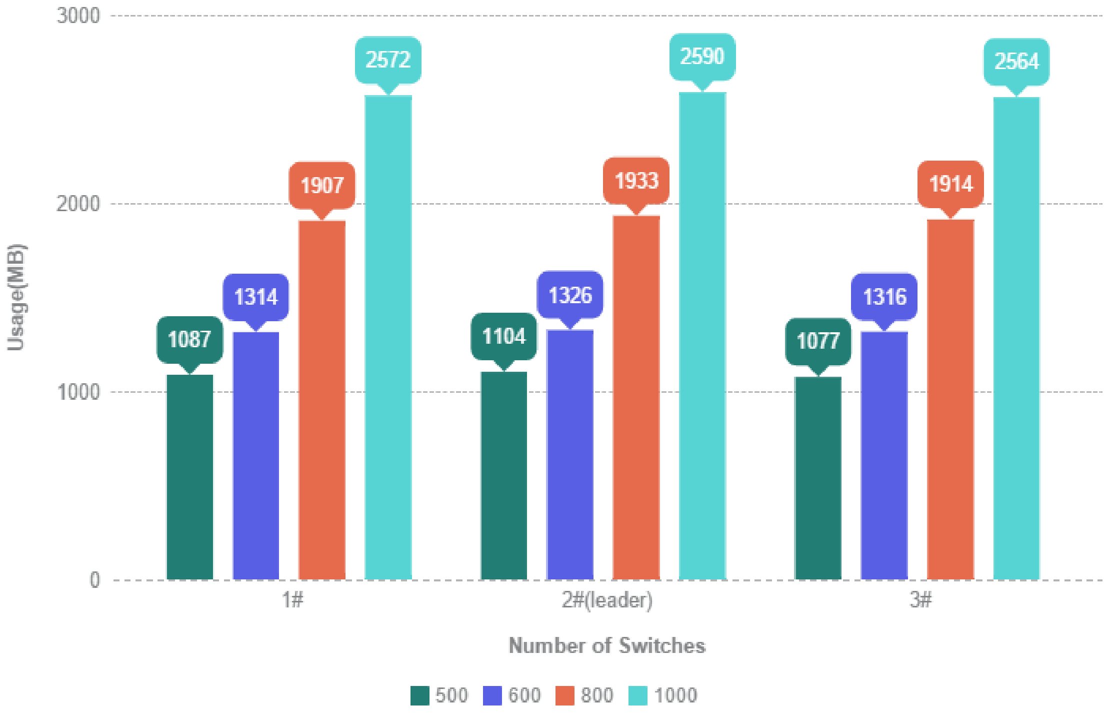


- Cluster Mode:
  In cluster mode, the memory usage of all the three controller instances should be monitored respectively. In this test, linear and ring topologies are implemented. As seen from the test results, even though there is a leader node, the memory usage of all the cluster nodes are approximately equal and slightly larger than standalone mode under the same circumstance. This is probably because of extra clustering and synchronization threads are running in each node and every node is required to have the entire information regarding the devices connected to the cluster. Due to this reason, the maximum number of control channels is not increased dramatically compares to standalone mode, but it does increase. Detailed results are shown below:

Memory Usage(LinearTopology)



Memory Usage(Ring Topology)



4.2. Topology Discovery Time

PURPOSE:

Benchmark the topology discovery time of controller for different kinds of network topologies and different node numbers.

PREREQUISITE

- The Controller under test should be able to run in both standalone mode and cluster mode.
- Controller under test must support topology discovery function without any unexpected errors.
- The default action for the table-miss entry in the switches must be output to controller or the switch has pre-installed flow entries for topology discovery messages.

RESULT EXPECTATION

- The topology discovery time of a controller cluster should be no longer than the time of the same controller running in standalone mode for the identical topology.
- The topology discovery time of the identical topology type which consists of less number of nodes should be no longer than the one consists of more number of nodes.
- The topology discovery time of different topology types consists of identical number of nodes may different from each other.

TOPOLOGY

In this test, it is recommended to setup different types of network topology with different node numbers. The topology figure shown below is a typical leaf-spine network setup in data center setting as an example.

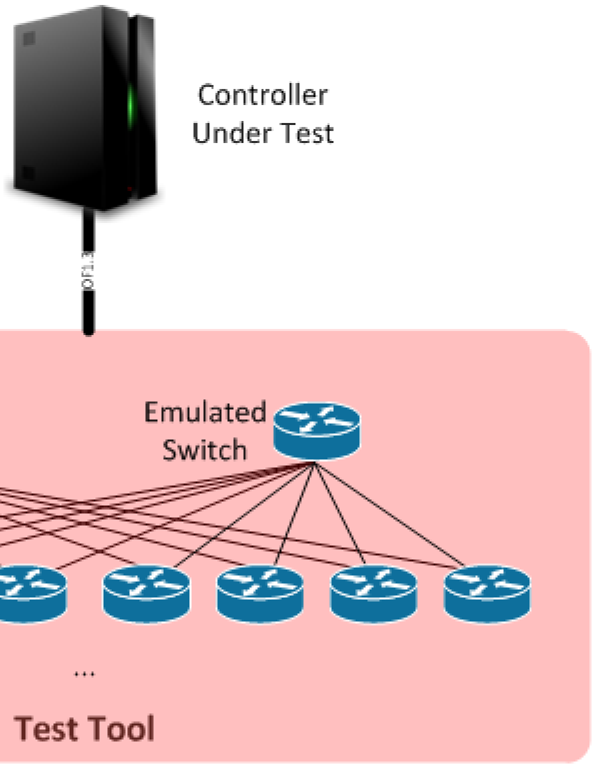


Fig 2.One Example for Topology Discovery Time test

METHODOLOGY

1. Start one controller running in standalone mode
2. Start capture
3. Switches start to establish connections to the controller.
4. Record the time(Tstart) for the controller starts topology discovery service
5. Record the time(Tfout) for the first discovery(LLDP) packet sent by controller via Packet_out messages
6. Record the time(Tfin) for the first Packet_in message received by controller
7. Record the time (Tlin) for the last Packet_in message sent by the emulated switches
8. Record the time(Tend) for the controller finalizes the topology discovery task
9. Get the topology discovery time T = Tend - Tstart
10. Also calculate Tfout - Tstart (**Tinit**)which is the initialization time of the controller topology discovery service, Tfin - Tfout(**Ttool**) and Tlin - Tfin(**Tnode**) which are the times may vary when using different test tools and different topology/node setups, Tend - Tlin(**Tproc**) is the time taken for controller to process all the information and generate the topology discovery result
11. Iterate this test for different topology types and number of nodes
12. Iterate this test for controller instances running in cluster mode

RESULTS

- Standalone Mode:
  Three different kinds of topologies are setup for the purpose of this test case: linear topology, ring topology and grid topology. According to the methodology, a graphical demonstration of Tinit, Ttool, Tnode and Tproc is given in Fig 3.

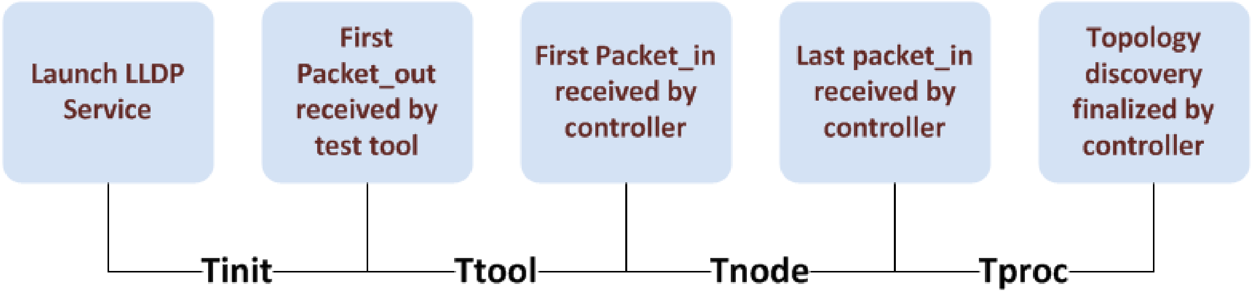
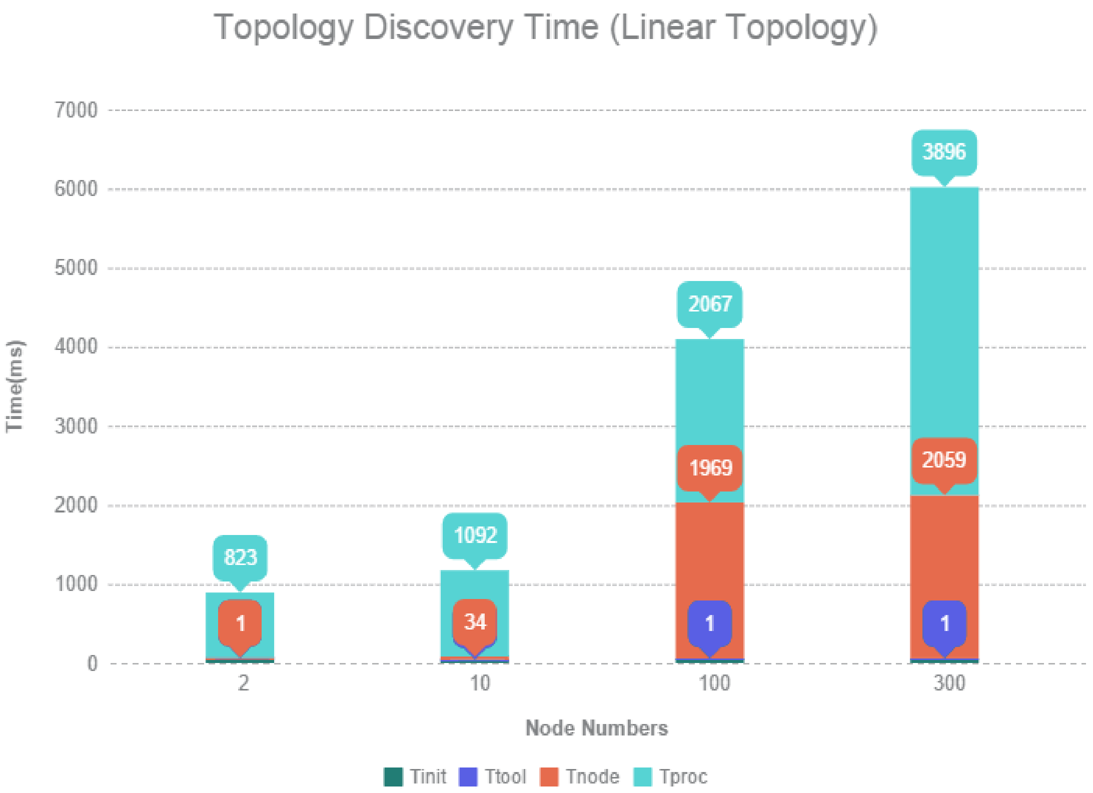
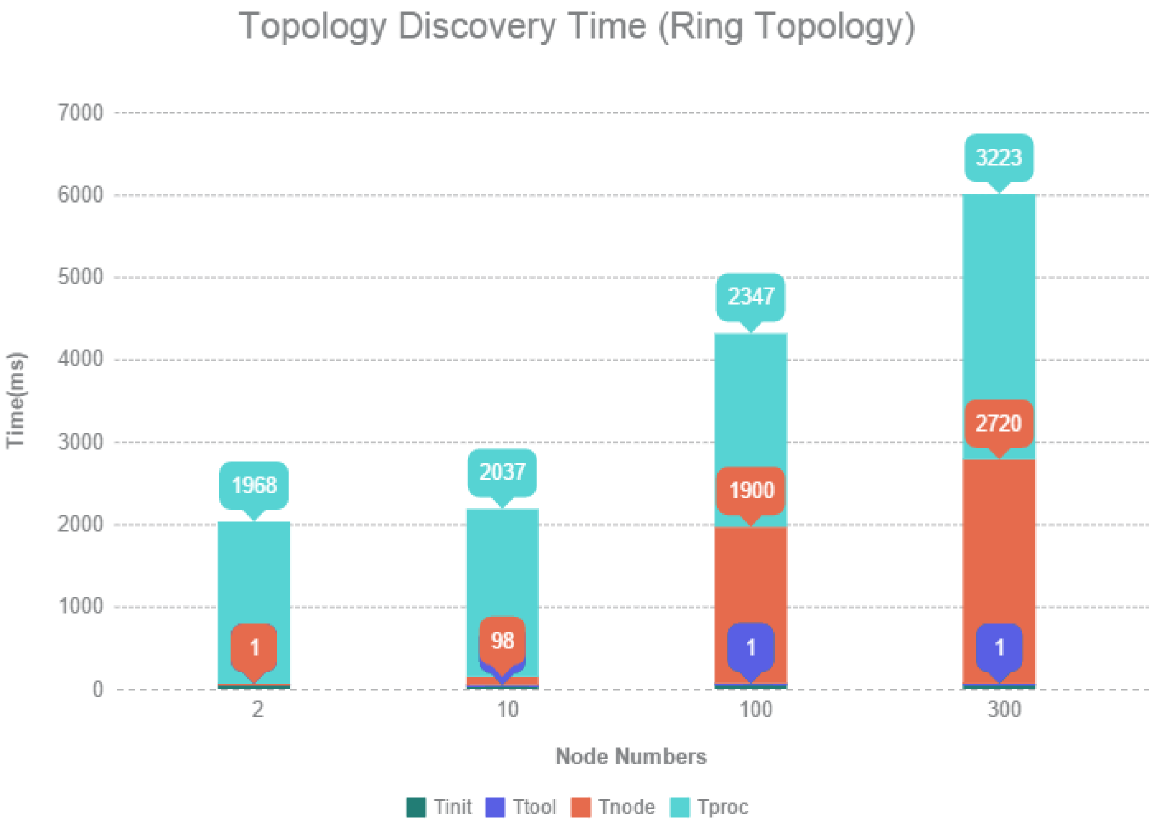
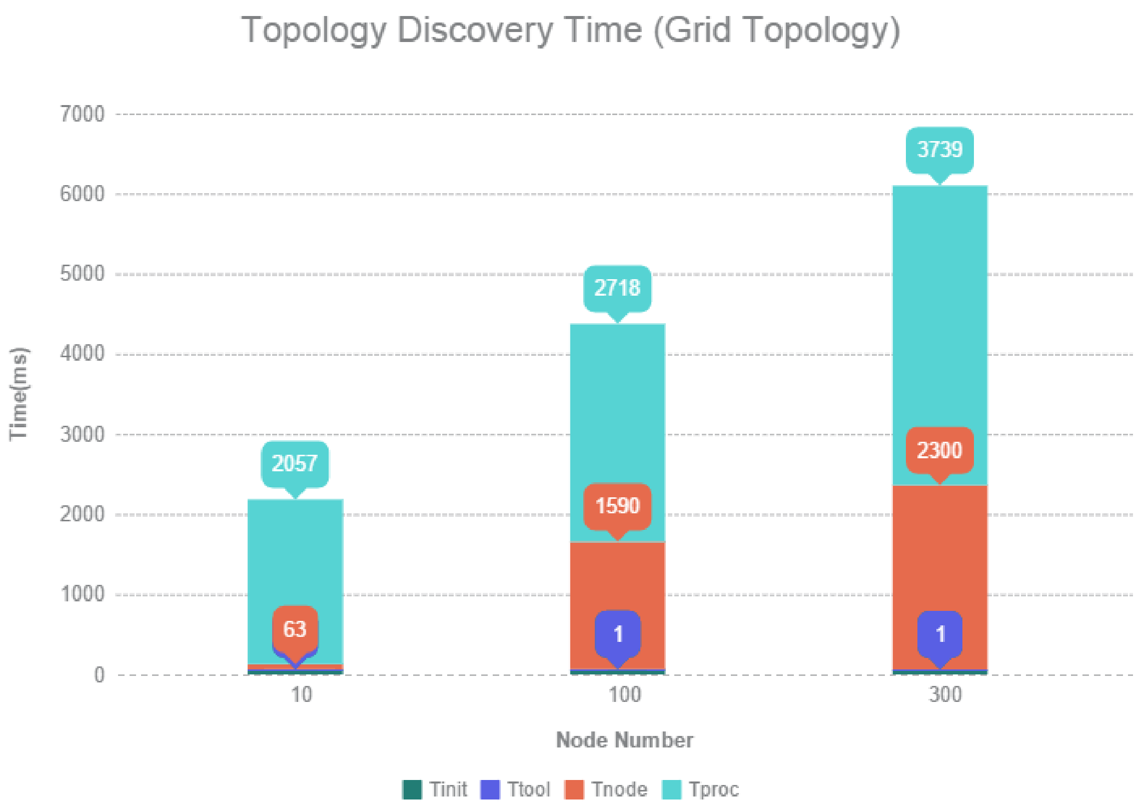

Launch LLDP Service
First Packet_out received by test tool
First Packet_in received by controller
Last packet_in received by controller
Topology discovery finalized by controller
Tinit
Ttool
Tnode
Tproc

Fig 3. Graphical demonstration of Tinit, Ttool, Tnode and Tproc

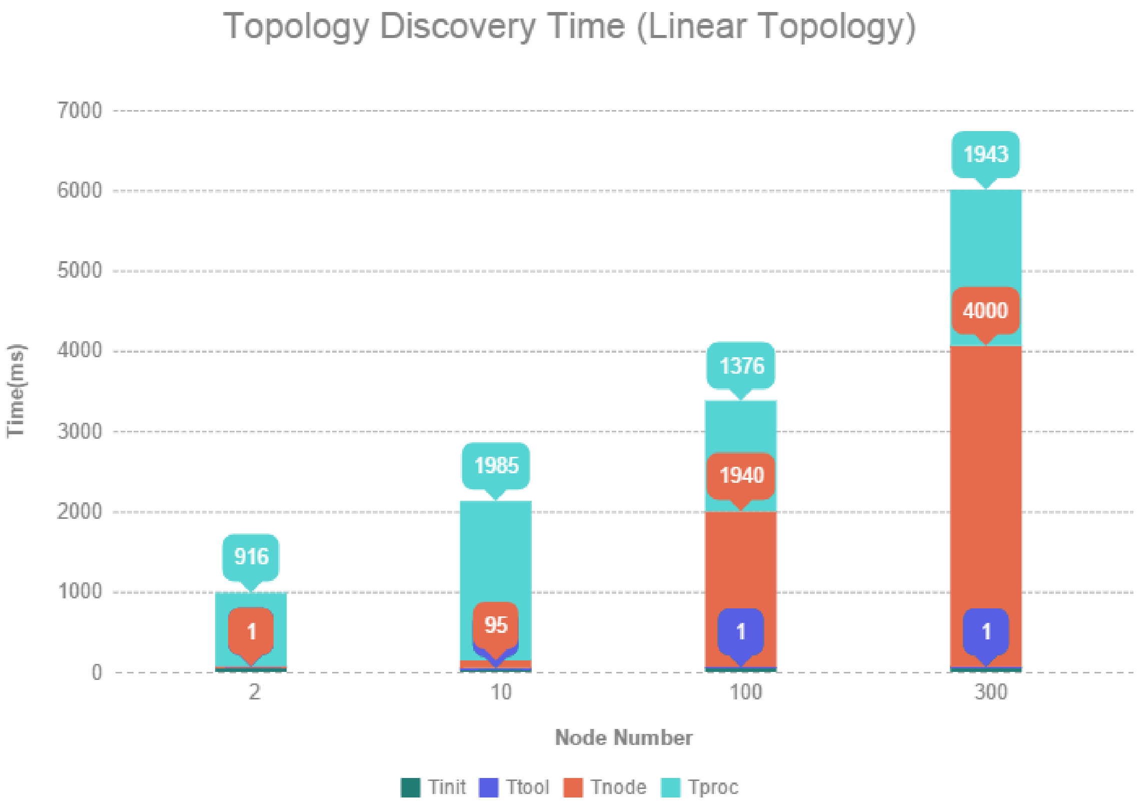
As seen in the results, one could tell the Tinit and Ttool remains relatively constant when the number of nodes varies. These two time frames are 60±5ms and 1ms respectively and cannot be easily seen in the chart when compare to Tnode and Tproc. Both of the Tnode and Tproc are directly proportional to the number of nodes and much larger than Tinit and Ttool. Detailed results are shown below, please note that the numbers are in stacked mode:


Topology Discovery Time (Linear Topology)
Time(ms)
7000
6000
5000
4000
3000
2000
1000
0
823
1
1092
34
2067
1969
1
3896
2059
1
2
10
100
300
Node Numbers
Tinit
Ttool
Tnode
Tproc

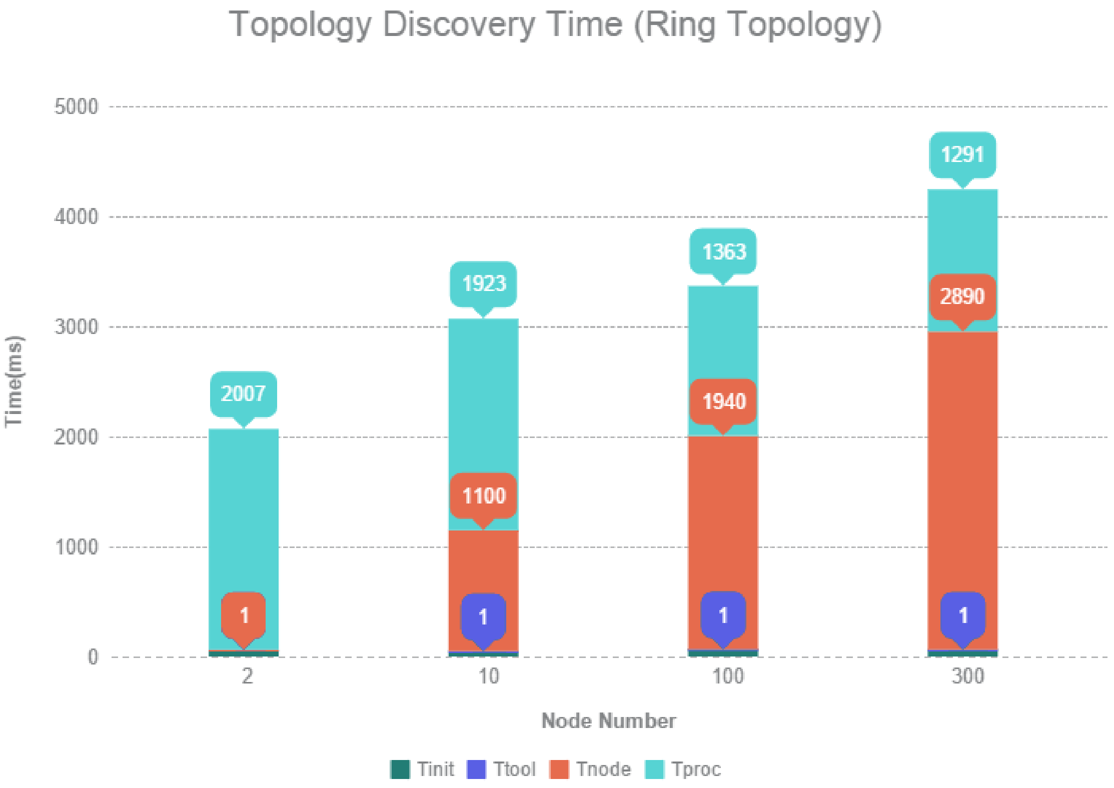

Topology Discovery Time (Ring Topology)
Time(ms)
7000
6000
5000
4000
3000
2000
1000
0
1968
1
2037
98
2347
1900
1
3223
2720
1
2
10
100
300
Node Numbers
Tinit
Ttool
Tnode
Tproc



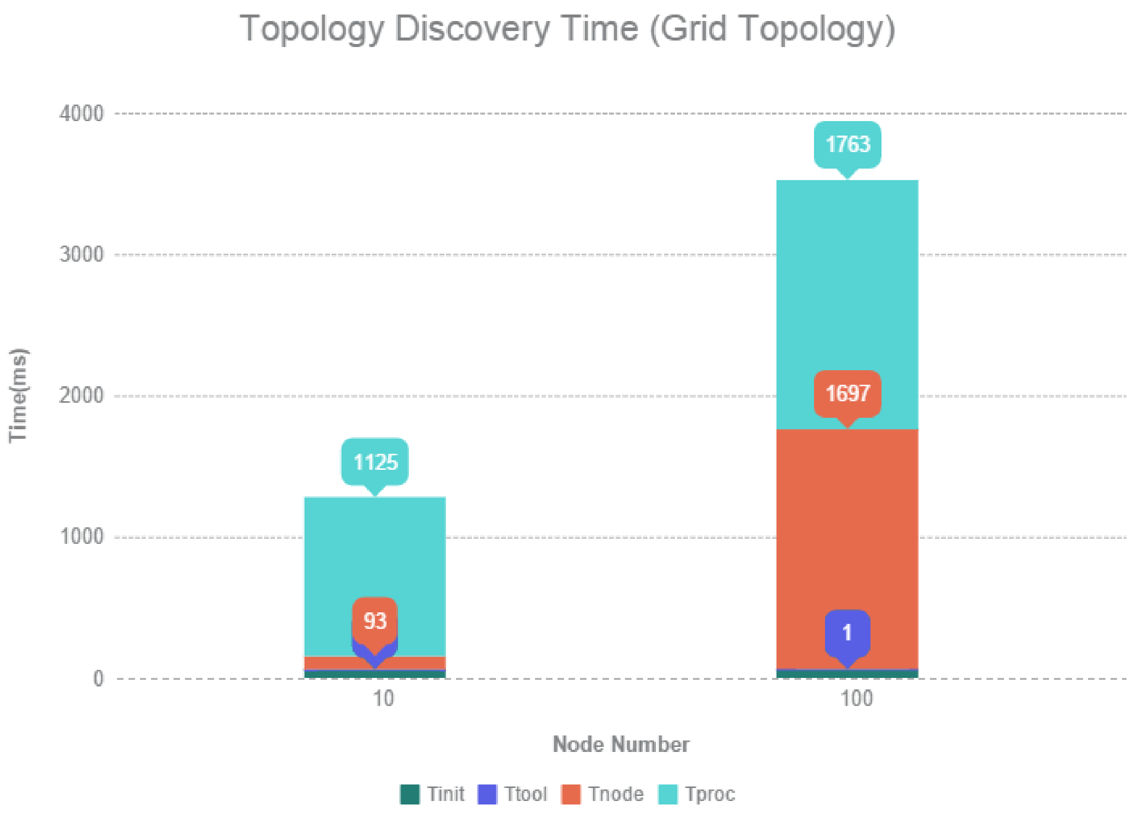
- Cluster Mode:
  In cluster mode, all the three kinds of topologies are implemented. Tinit and Ttool are identical to the tests implemented in standalone mode. When the node number is relatively small, the Tproc time is longer than the standalone mode because of the synchronization overhead, but when the number is relatively large(100 or 300), the Tproc decreases because of the increased process power of a cluster becomes a more dominated factor. According to the captures, only the leader node sends LLDP packets via Packet_out messages and the switches send replicated Packet_in messages back to all the three nodes. Because of this reason, Tnode increases in cluster mode. Therefore the overall topology discovery time may increase and may also possibility decrease as well. According to the results, the overall time is heavily depends on the type of topology. And the conclusions above are not applied to all of the three types. More research is required in this area. Similar to previous test case, overhead of synchronization will be added for sure but consider the small number of the controller instances, this should not be a heavy thread. Detailed results are shown below:



Topology Discovery Time (Linear Topology)



Topology Discovery Time (Ring Topology)



4.3. *Topology Change Events Response Time*

PURPOSE:

Benchmark the response time of controller under test with topology change events. These events include port up/down and switch up/down. The word 'response' not only means the detection of the events, but also the entire topology update activities.

PREREQUISITE

- The Controller under test should be able to run in both standalone mode and cluster mode.
- Controller under test must support topology discovery function without any unexpected errors.
- The default action for the table-miss entry in the switches must be output to controller or the emulated switch has pre-installed flow entries for topology discovery messages.
- The topology change events involved with this test are: port up/port down/switch up/ switch down.

RESULT EXPECTATION

- The topology events response time of a controller cluster should be no longer than the time of the same controller running in standalone mode for the identical topology and events.
- The topology events response time of the same topology type consists of less number of nodes should be no longer than the one consists of more number of nodes.
- The topology events response time of different topology types consists of identical number of nodes may different from each other.

- The port/switch down events should be responded faster than port/switch up events since the former events have greater impact on current topology and convergence time.

Topology

In this test it is recommended to setup different types of network topology with different node numbers. The topology figure shown below is a typical leaf-spine network setup in data center setting as an example.

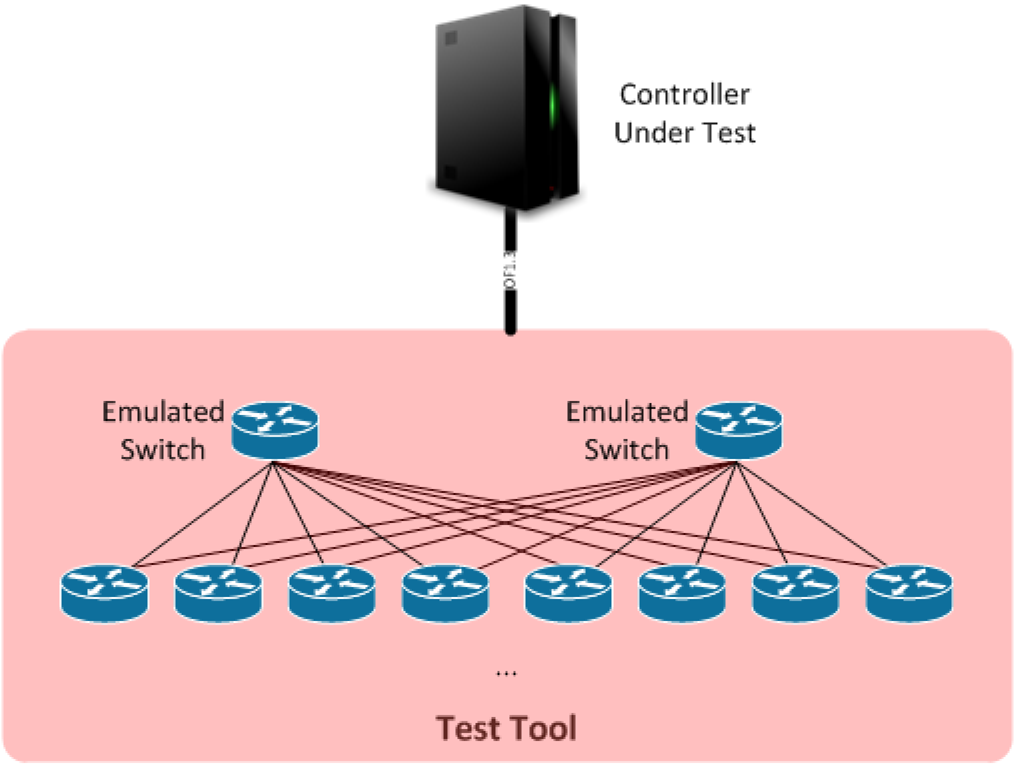


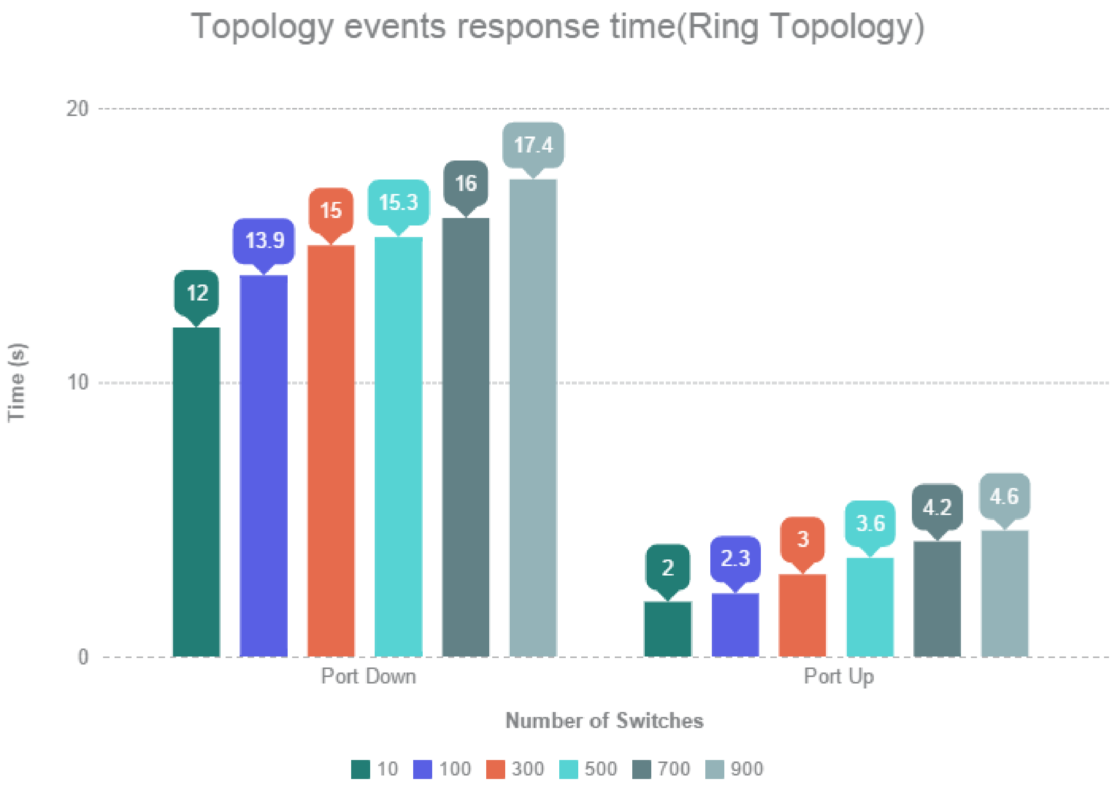
Fig 4.One Example for Topology Change Event Response Time test

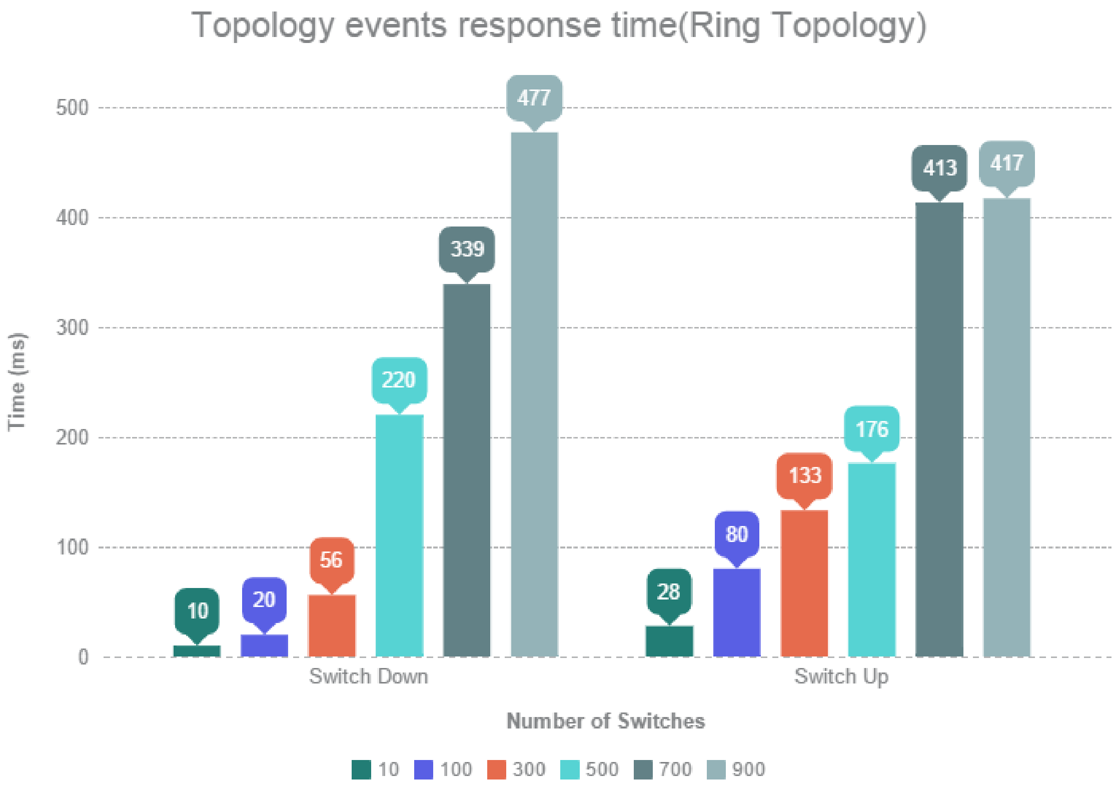
Methodology

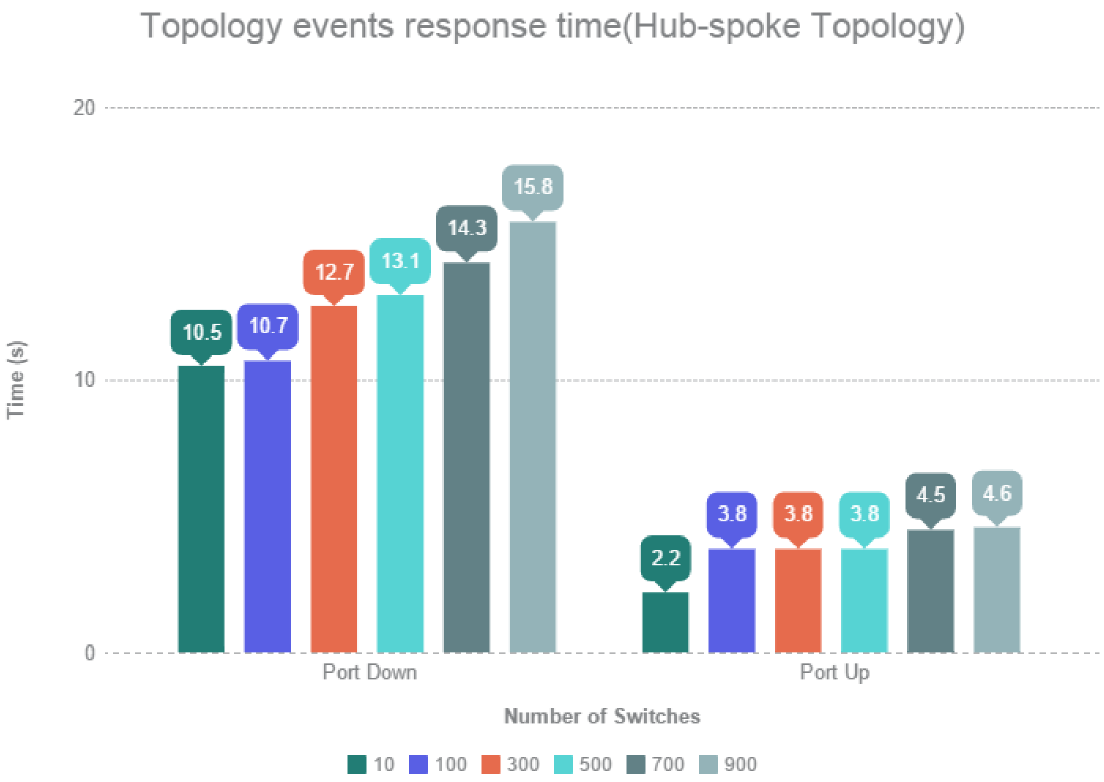
1. Start one controller running in standalone mode
2. Start capture
3. Switches start to establish connections to the controller
4. Wait until the topology discovery completed
5. Set one port of an emulated switch down manually
6. Record the time(T1) for the notification message received by the controller
7. Record the time(T2) for the controller to finalize the topology re-discovery
8. Get the topology events response time T = T2 - T1
9. Iterate this test for different topology types and number of nodes
10. Iterate this test for different kinds of topology events described in the prerequisite section
11. Iterate this test for controller instances running in cluster mode

Results

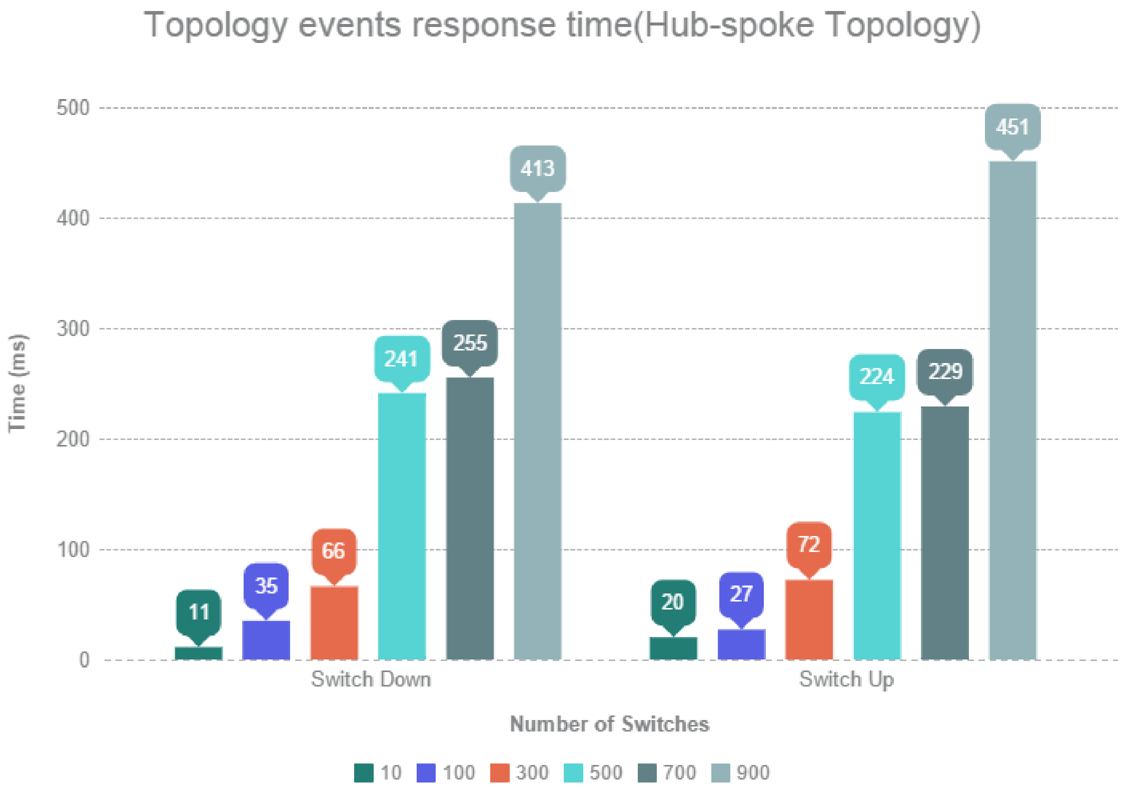
- Standalone Mode:
  This test case is implemented by two kinds of topologies: ring topology and hub-spoke topology. It seems that the switch up/down events are much easier for the controller to deal with than the port up/down events. The handling of port down event is slower than the port up event which is opposite to the result expectation. The time taken for port down event is longer than 10 seconds which indicates a plenty of room for improvement for the controller. Also please note that in the charts below, the time unit is second for port down/up events and millisecond for switch down/up events. Detailed results are shown below:


Topology events response time(Ring Topology)
Time (s)
20
10
0
12
13.9
15
15.3
16
17.4
2
2.3
3
3.6
4.2
4.6
Port Down
Port Up
Number of Switches
10
100
300
500
700
900


Topology events response time(Ring Topology)
Time (ms)
500
400
300
200
100
0
10
20
56
220
339
477
28
80
133
176
413
417
Switch Down
Switch Up
Number of Switches
10
100
300
500
700
900

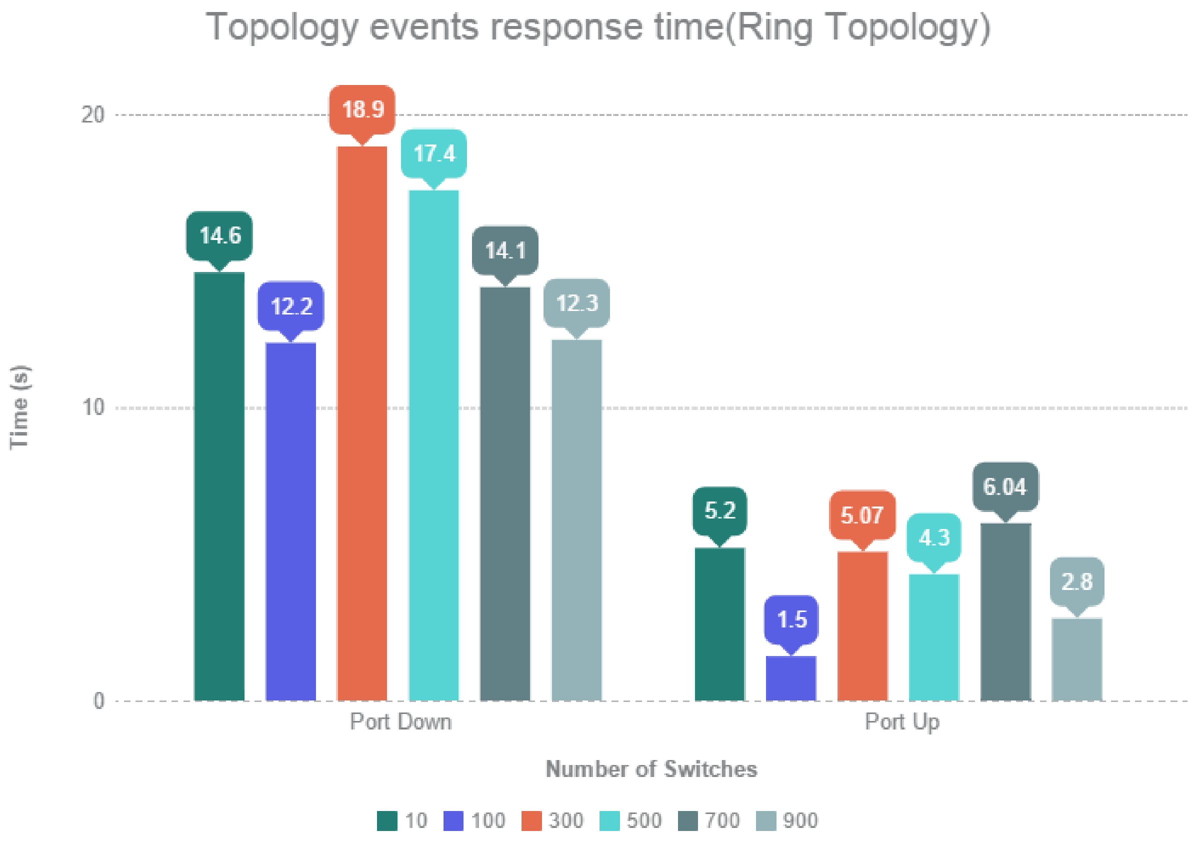


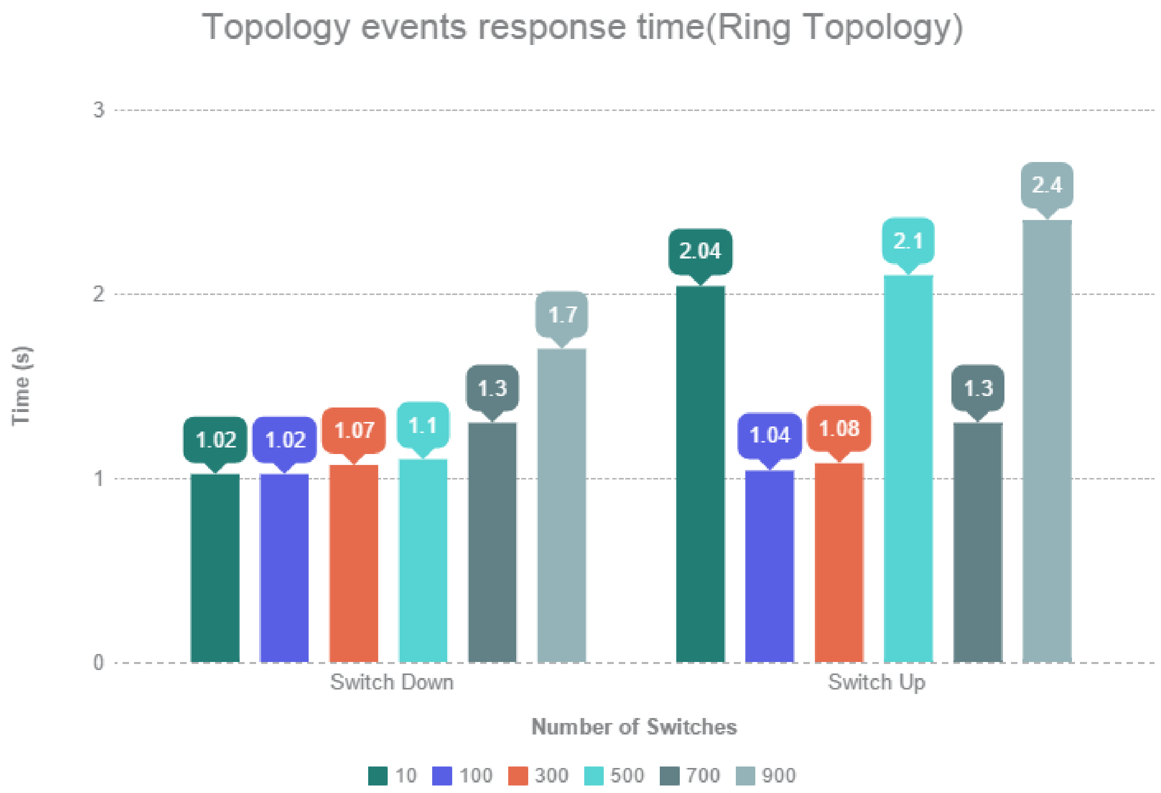
Topology events response time(Hub-spoke Topology)

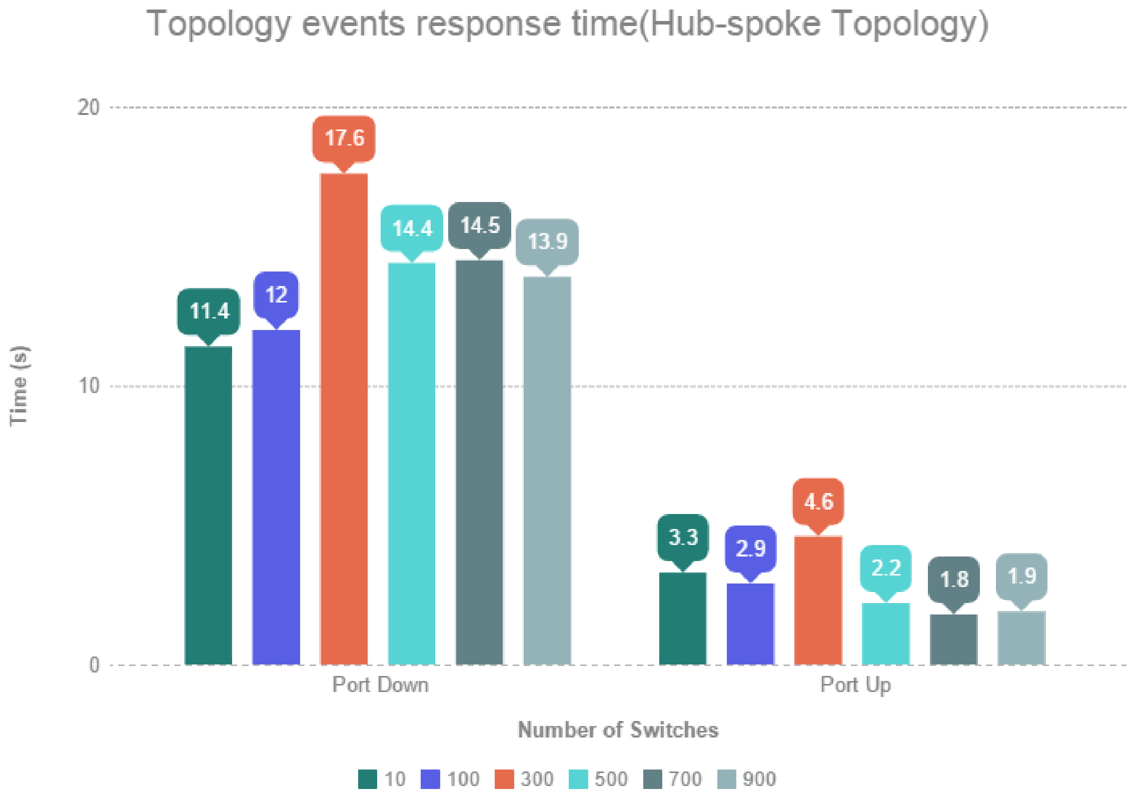


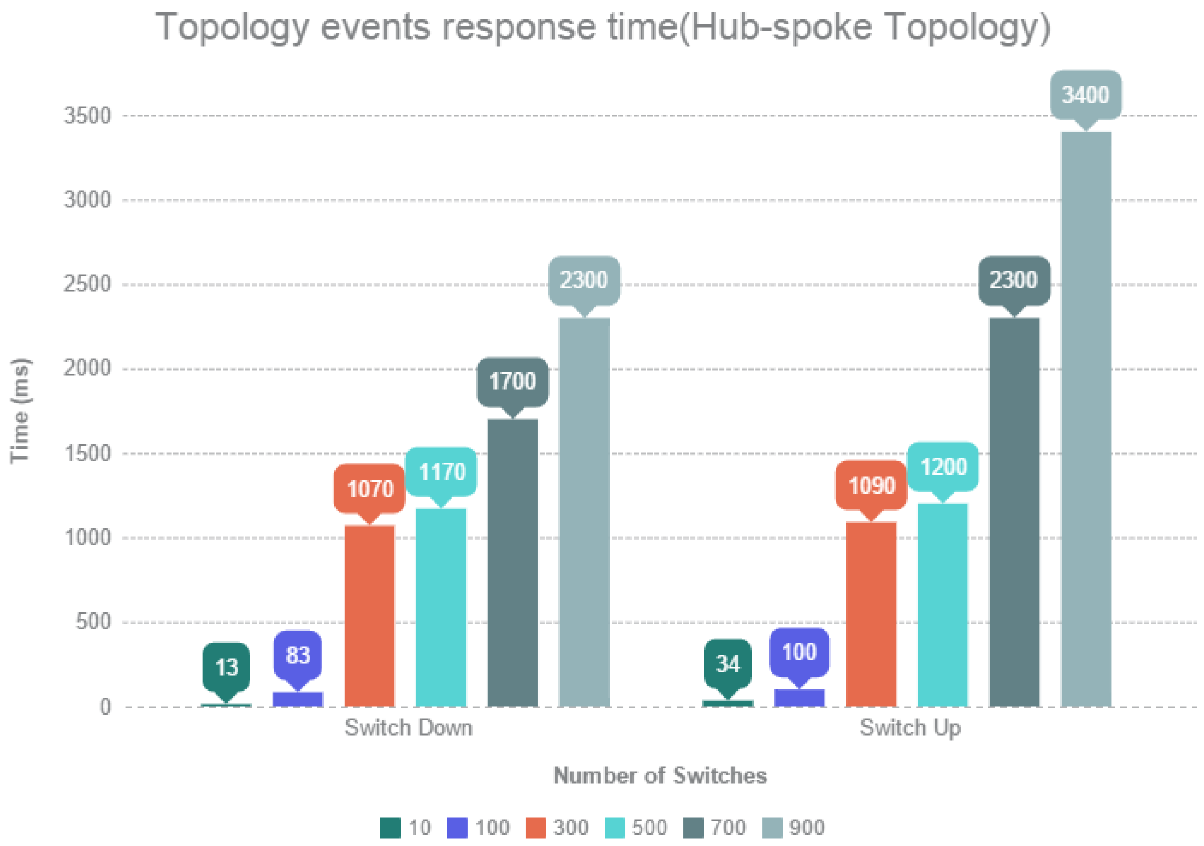
Topology events response time(Hub-spoke Topology)

- Cluster Mode:
  In cluster mode, three types of topology are implemented. As seen from the results, the performance behavior regarding the topology change events is not predictable any more in cluster mode. The synchronization process, topology type and internal topology discovery mechanism (the time interval for sending LLDP packets), all of these influential factors make the results stochastic. One cannot assert that cluster is 'faster' than standalone or less switch nodes is easier to handle. This may be the real situation a SDN network user may face during the deployment process of his network and it definitely need to be improved. Also please pay attention to the unit of the Y-axis when you are reviewing the results. Detailed results are shown below:




Topology events response time(Ring Topology)
Time (s)
3
2
1
0
1.02
1.02
1.07
1.1
1.3
1.7
2.04
1.04
1.08
2.1
1.3
2.4
Switch Down
Switch Up
Number of Switches
10
100
300
500
700
900


Topology events response time(Hub-spoke Topology)
Time (s)
20
10
0
11.4
12
17.6
14.4
14.5
13.9
3.3
2.9
4.6
2.2
1.8
1.9
Port Down
Port Up
Number of Switches
10
100
300
500
700
900



4.4. Reactive Flow Setup Rate

PURPOSE:

Benchmark the rate of controller under test to setup flows in switches reactively.

PREREQUISITE

- The controller under test should be able to run in both standalone mode and cluster mode.
- Controller under test must support topology discovery function without any unexpected errors.
- The default action for the table-miss entry in the switches must be output to controller.
- An application should run above the controller in order to react to the flow setup requests at the maximum possible rate.
- No buffer is required in the switch.

RESULT EXPECTATION

- The reactive flow setup rate of a controller cluster should be no less than the rate of the same controller running in standalone mode.
- The reactive flow setup rate should increase along with the Packet_in messages sending rate of the emulated switch within a certain range.

TOPOLOGY

In this test only one emulated switch is connected to the controller under test. A traffic generator connected with the switch is trying to generate APR_request and reply messages so that the switch could send these messages to controller via Packet_in messages. The controller will process the Packet_in messages and response with Flow_add messages in order to create bidirectional connections.

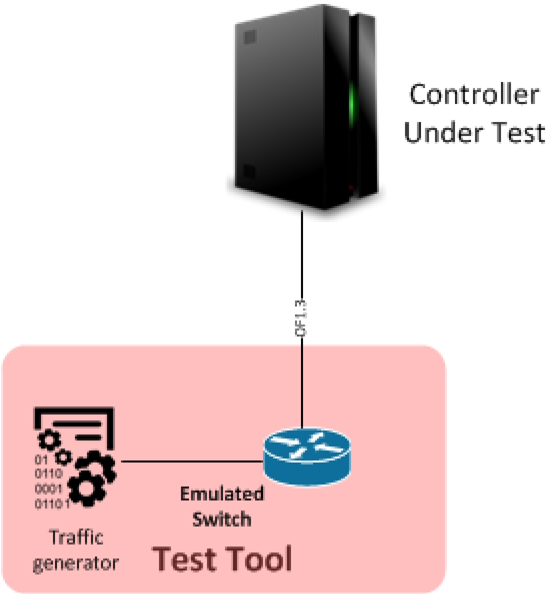


Fig 5.Topology for Reactive Flow Setup Rate test

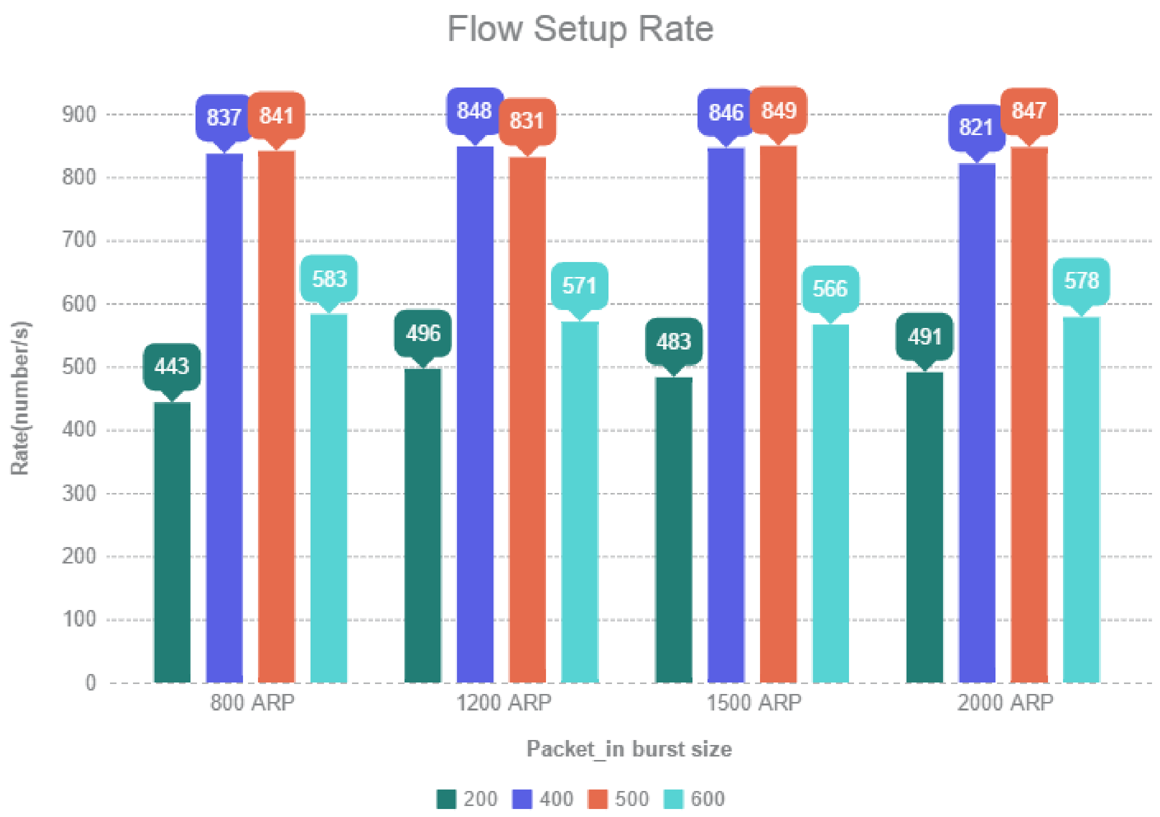
METHODOLOGY

1. Start one controller running in standalone mode
2. Start capture
3. Switch(es) start to establish connection(s) to the controller
4. Wait until the topology discovery completed
5. Controller starts a simple L2 learning switch application
6. A certain number of ARP_request messages are sent to the controller via Packet_in messages
7. After the Packet_out messages sent out by the controller, test tool starts sending corresponding APR_reply messages(for the learning switch application) via Packet_in messages at a certain TX burst size(numbers of Packet_in messages per second)
8. Record the time Tf of the first Flow_mod message received by switch
9. Record the time Tl of the last Flow_mode message received by switch
10. Ensure that all the ARP_reply Packet_in messages have been responded by a Flow_mod message
11. Reactive flow setup rate R = total amount of Flow_mod message / (Tl - Tf)
12. Iterate this test for different Packet_in burst size
13. Iterate this test for controller instances running in cluster mode

RESULTS

- Standalone Mode:

  One switch is setup for the purpose of this test case. The test tool is configured for different total ARP_request/reply number and burst size for sending Packet_in messages. For some particular configurations, the controller is not able to response all the ARP_reply messages by adding two bidirectional flow entries, but the actual number of Flow_add messages is still counted and treated as a valid number of the test. From the result, one could tell that when the burst size is lower than a certain threshold, the flow setup rate is directly proportional to the burst size but when the burst size exceeds the threshold, the flow setup rate goes down mainly because the controller is not able to response to all the Packet_in messages. The total number of APR messages has little influence on the results. Detailed results are shown below:



- Cluster Mode:
  In cluster mode, each Packet_in message containing the ARP_request/reply message is replicated and sent to all of the three controller instances by the switch. Each ARP_reply message is supposed to be responded with two corresponding Flow_mod messages in order to setup bidirectional connections. During the result analysis process of this test case, it is surprised to find that two or three instances are sending identical Flow_add messages to the emulated switch. In other words, a part of ARP_reply messages are responded with four or six Flow_mod messages for each. This is probably because the synchronization process does not perform a good coordination work under this circumstance. According to the OpenFlow 1.3.4 specification, if the CHECK_OVERLAP flag in this Flow_mod message is set, this will cause the switch to generate error messages. Because of this reason, the result is treated as invalid so there is no chart for this test.

4.5. Reactive Packet_out Response Rate

PURPOSE:

Benchmark the rate for controller under test to response Packet_in messages with Packet_out messages.

PREREQUISITE

- The controller under test should be able to run in both standalone mode and cluster mode.
- Controller under test must support topology discovery function without any unexpected errors.
- The default action for the table-miss entry in the switches must be output to controller.

- An application should run above the controller in order response Packet_in message with Packet_out messages.
- No buffer is required in the switch.

Result Expectation

- The reactive Packet_out response time of a controller cluster should be no longer than the time of the same controller running in standalone mode.

Topology

In this test only one emulated switch is connected to the controller under test. A traffic generator connected with the switch is trying to generate APR_request messages so that the switch could send these messages to controller via Packet_in messages. The controller will process the Packet_in messages and response with Packet_out messages in order to flood the ARP_request.

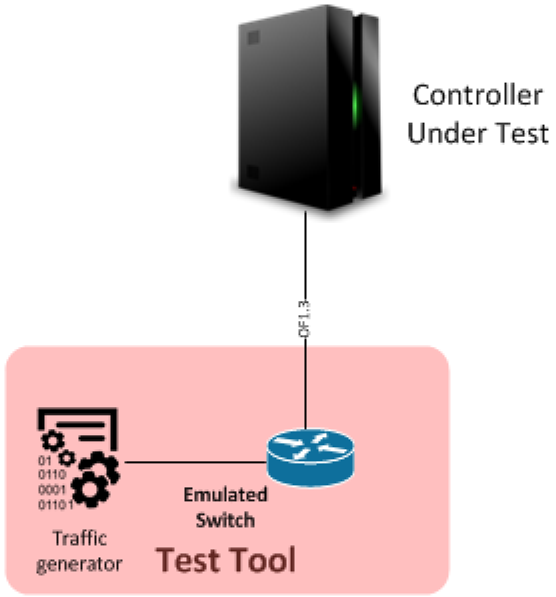


Fig 6.Topology for Reactive Packet_out Response Time test

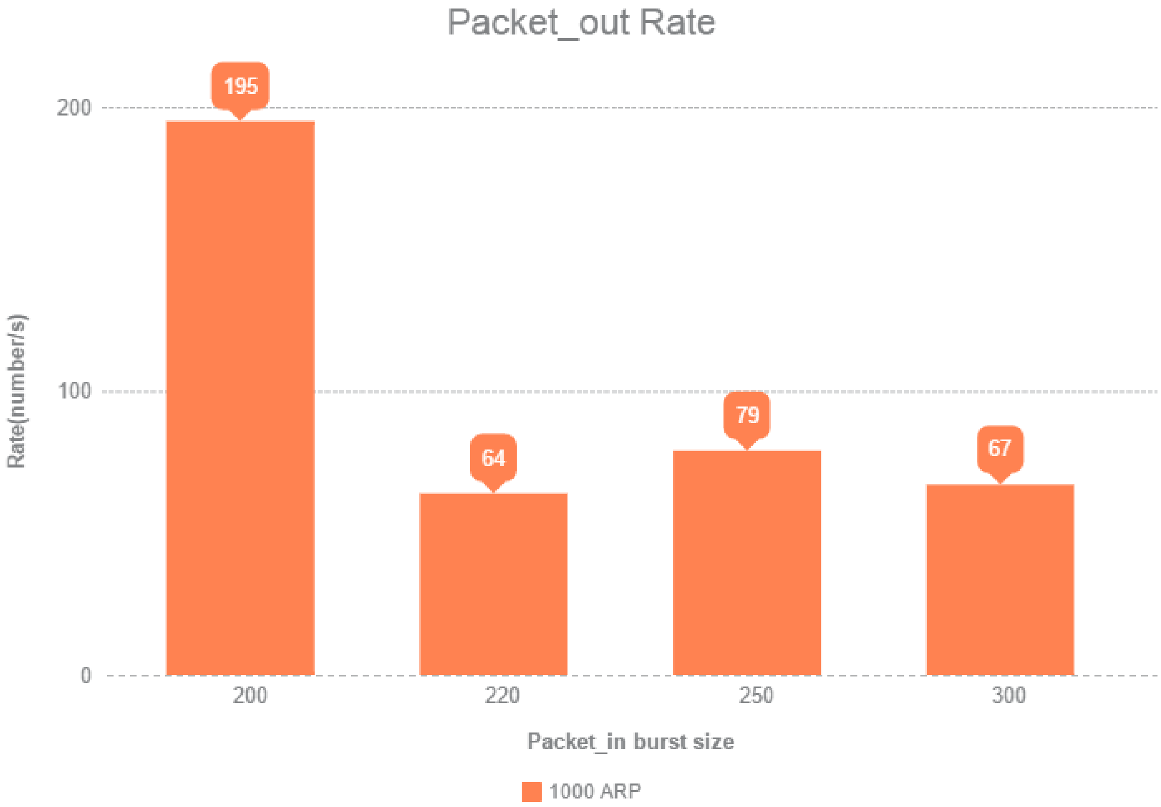
Methodology

1. Start one controller running in standalone mode
2. Start capture
3. Switch start to establish connections to the controller
4. Wait until the topology discovery completed
5. Controller starts L2 learning switch application
6. Switch send ARP_request messages to controller via Packet_in messages at a certain Packet_in TX burst size(number of Packet_in messages per second)
7. Record the time T1 when controller receives the first Packet_in message contains the first packet
8. Record the time T2 when the last Packet_out message received by switch
9. Ensure all the ingress packets are sent out by Packet_out messages
10. Packet_out response rate R = Total amount of Packet_in messages / (T2 - T1)
11. Iterate this test for different Packet_in TX burst size
12. Iterate this test for controller instances running in cluster mode

Results

- Standalone Mode

One switch is setup for the purpose of this test case. The total number of APR_request messages to be sent remains constant (1K) while applying different TX burst size. Unlike the previous reactive flow setup rate test case, the Packet_out rate does not increase along with the TX burst size. According to the traffic captures, quite a few of TCP retransmissions take place for the Packet_out messages when the burst size exceeds 200 and has a significant negative influence on the rate. Even though, the actual number of Packet_out messages sent by controller is still counted and treated as valid data of the test case. Detailed results are shown below:



- Cluster Mode
  In cluster mode, the situation is similar to the previous test case. Each Packet_in message containing the ARP_request message is replicated and sent to all of the three controller instances by the switch. Each ARP_request message is supposed to be responded with one Packet_out message in order to flood the ARP messages. It is found that more than one instance are sending identical Packet_out messages to the emulated switch. In other words, the total amount of Packet_out messages sent by controller is larger (approximately 1.5 times) than the total ARP_request messages. This is also highly likely because that the synchronization process does not perform a good coordination work under this circumstance. Because of this reason, the result is treated as invalid so there is no chart for this test.

4.6. Reactive Path Setup Time

PURPOSE:

Benchmark the time of a controller under test to setup an end-to-end path for different kinds of topologies and number of nodes.

PREREQUISITE

- The controller under test should be able to run in both standalone mode and cluster mode.
- Controller under test must support topology discovery function without any unexpected errors.
- The default action for the table-miss entry in the switches must be output to controller.

- An application should run above the controller in order to setup a end-to-end path reactively at the fastest speed.
- A loop prevention mechanism is enabled.
- The controller should be able to obtain the knowledge about the locations of the end point hosts.

RESULT EXPECTATION

- The reactive path setup time of a controller cluster should be no longer than the time of the same controller running in standalone mode.
- The reactive path setup time of the same topology type consists of less number of nodes should be no longer than the one consists of more number of nodes.
- The reactive path setup time of different topology types consists of identical number of nodes may different from each other.

TOPOLOGY

In this test it is recommended to setup different types of network topology with different node numbers. The topology figure shown below is a typical leaf-spine network setup in data center setting as an example. For each configuration, the two hosts are setup at the locations which could have longest possible distance.

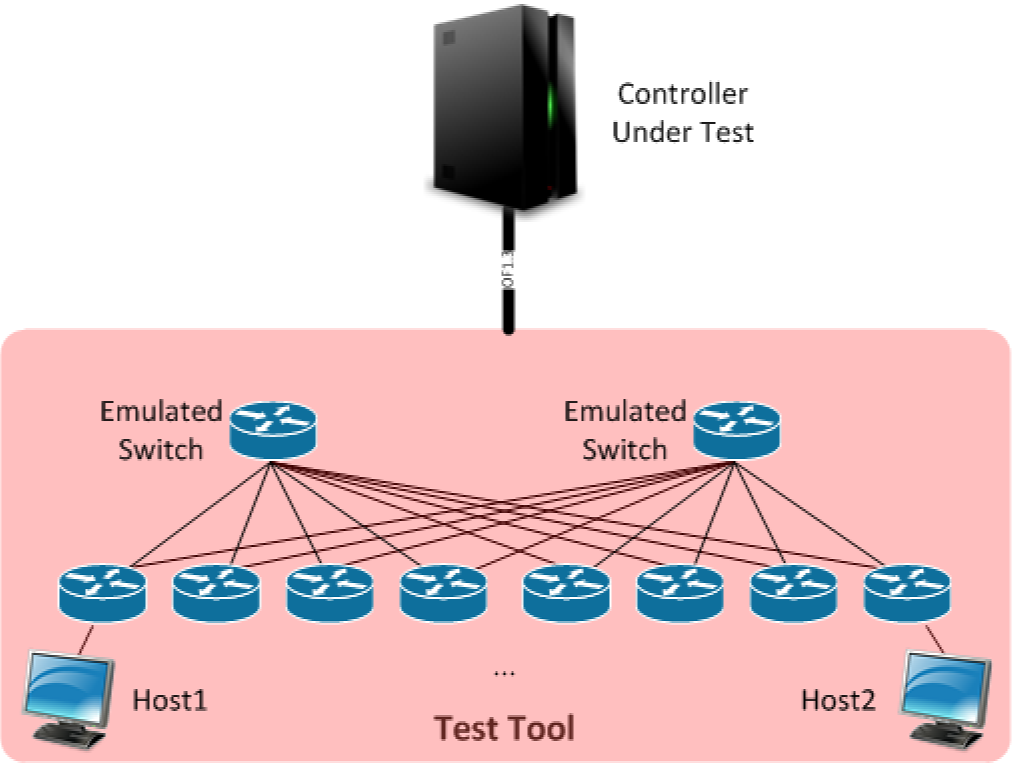


Fig 7.An Example for Reactive Path Setup Rate test

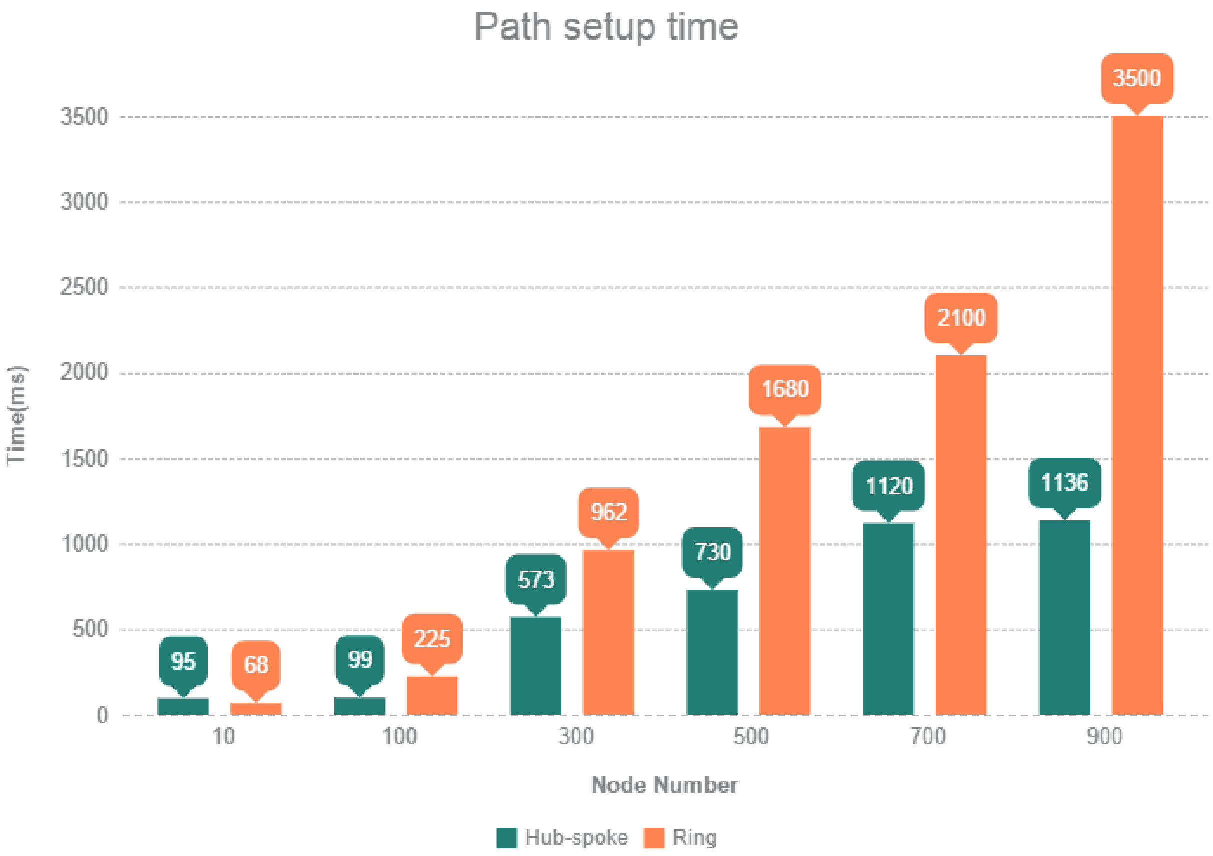
METHODOLOGY

1. Start one controller running in standalone mode
2. Start capture
3. Switches start to establish connections to the controller
4. Wait until the topology discovery completed
5. Controller starts L2 learning switch application
6. Record the time T1 when controller receives the Packet_in message contains ARP_request packet
7. Controller processes the Packet_in message and install flow entries to the switches
8. Record the time T2 when the last Flow_mod messages received by switches
9. Host1 sends a packet to Host2 to ensure there is no Packet_in message received by controller anymore
10. Reactive path setup time T = T2 - T1

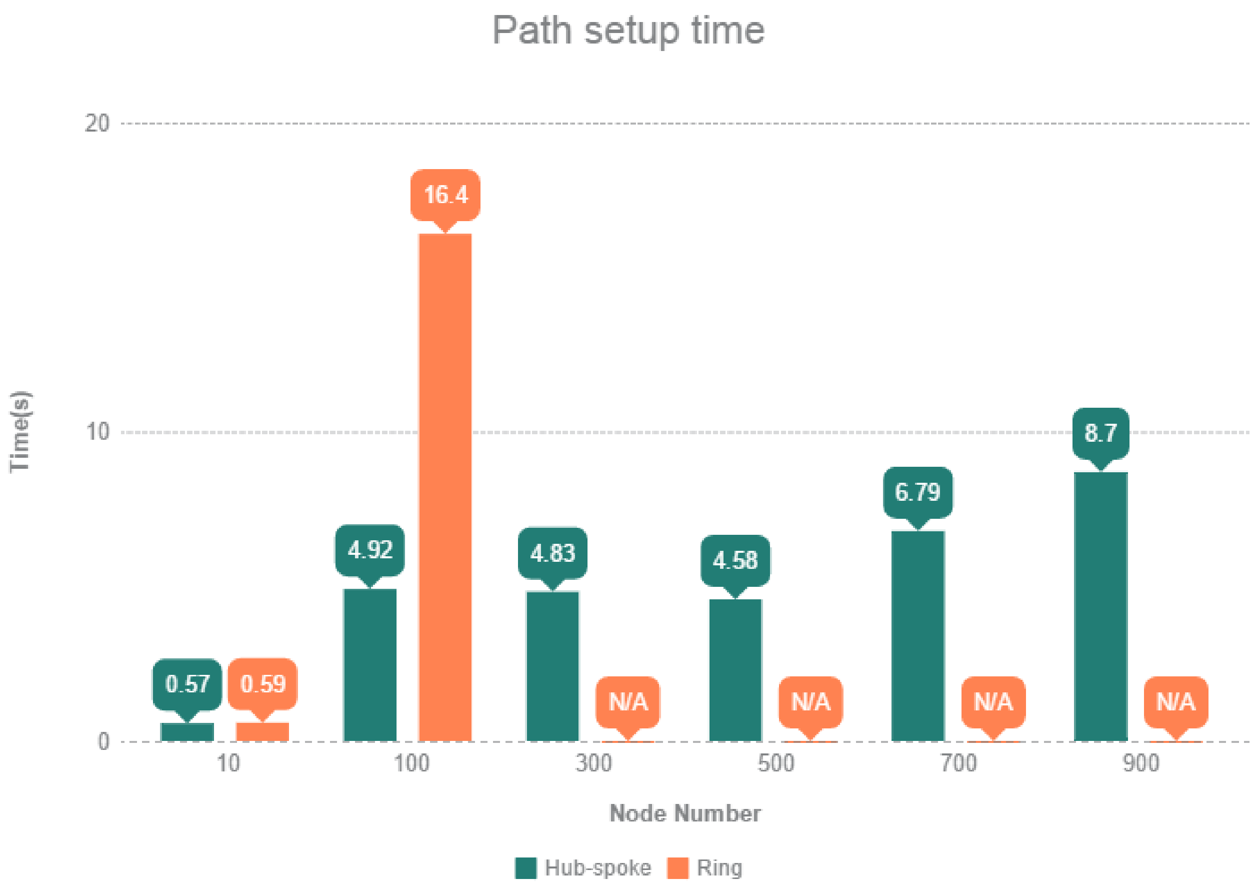
11. Iterate this test for different topology types and number of nodes
12. Iterate this test for controller instances running in cluster mode

RESULTS

- Standalone Mode:
  This test case is implemented by two kinds of topologies: ring topology and hub-spoke topology. The results seems quite logical, the path setup time is directly proportional to the number of nodes in the topology. For different types of topologies, the increase rate (Time / Node Number) is not identical. The setup time increases dramatically for ring topology where the number of nodes increases from 700 to 900. Detailed results are shown below:



- Cluster Mode:
  In cluster mode, two types of topologies are implemented. In all the configurations, host1 and host2 are able to reach each other after the cluster finishes installing flow entries. During the result analysis process, it is found that many unnecessary or identical flow entries are installed when the node number is larger than 300 in ring topology. The situation is quite similar to Test Case 4.4 and 4.5. According to the discussion in these two test cases, the result is treated as invalid and marked as N/A in the chart. For Hub-spoke topology, the number of flow entries is correct but the time for path setup increases significantly when compares to the standalone mode. Please note that the unit of time is second for cluster mode and millisecond for standalone mode. Detailed result is shown below:



5. Extra Performance Test Cases Introduction

In this section some extra performance test cases are introduced. The test results are not included in this white paper while the test methodologies are well explained. We are seeking for more collaboration from test tool vendors and controller vendors for better implementation. Results of these test cases will be included in next release of controller performance test white paper.

5.1. Proactive Flow Setup Rate

Purpose:

Benchmark the rate of controller under test to setup flows in switch proactively.

Prerequisite

- The controller under test should be able to run in both standalone mode and cluster mode.
- Controller under test must support topology discovery function without any unexpected errors.
- The default action for the table-miss entry in the switches must be output to controller.
- An application should run above the controller in order to install flow entries to switch proactively at the maximum possible rate.

Result Expectation

- The proactive flow setup rate of a controller cluster should be no less than the rate of the same controller running in standalone mode.
- The proactive flow setup rate should increase along with the number of switch nodes existing in the environment within a certain range.

TOPOLOGY

This topology sets one node as an example. Normally, different number of nodes will influence on the results, therefore more emulated switches in this topology is recommended.

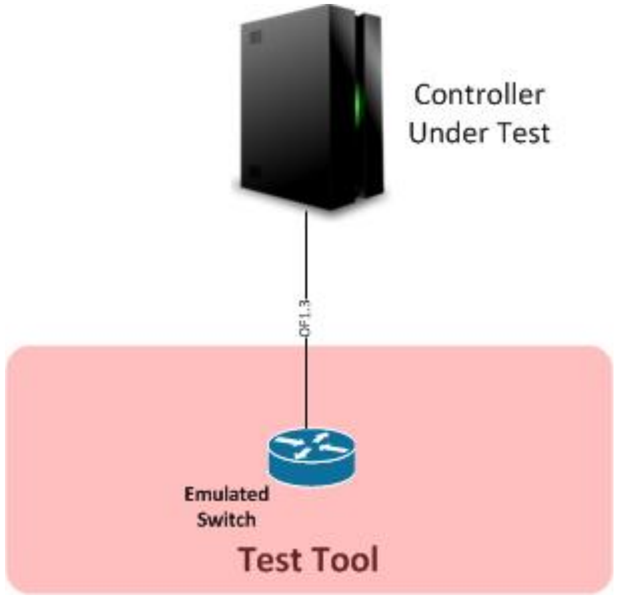


Fig 8.Topology for Proactive Flow Setup Rate test

METHODOLOGY

1. Start one controller running in standalone mode
2. Start capture
3. Switch(es) start to establish connections to the controller
4. Wait until the topology discovery completed
5. Controller starts an application which could install flow entries at the maximum rate proactively to the switch
6. Record the time Tf of the first Flow_mod message received by switch
7. Record the time Tl of the last Flow_mode message received by switch
8. Proactive flow setup rate R = amount of Flow_mod messages / (Tl - Tf)
9. Iterate this test for different number of switch nodes
10. Iterate this test for controller instances running in cluster mode

REMARKS

If the controller just installs meaningless flow entries to the switch, the result is not applicable for the users. This test case could be a way to test the latency of the controller’s northbound interface APIs. A time synchronization mechanism is required between the controller under test and the test tool.

5.2. Reactive Path Failure Recovery Time

PURPOSE:

Benchmark the time of a controller under test to setup an alternative end-to-end path when the previous one fails.

PREREQUISITE

- The controller under test should be able to run in both standalone mode and cluster mode.
- Controller under test must support topology discovery function without any unexpected errors.
- The network must have redundant paths between source and destination ends.
- An application should run above the controller in order to detect the failure and setup an alternative end to end path reactively at the fastest possible speed.
- A loop prevention mechanism is enabled.

- The controller should be able to obtain the knowledge about the locations of the end point hosts.

Result Expectation

- The reactive path failure recovery time of a controller cluster should be no longer than the time of the same controller running in standalone mode.
- The reactive path failure recovery time of the same topology type consists of less number of nodes should be no longer than the one consists of more number of nodes.
- The reactive path setup time of different topology types consists of identical number of nodes may different from each other.

Topology

In this test it is recommended to setup different types of network topology with different node numbers. The topology figure shown below is a typical leaf-spine network setup in data center setting as an example. For each configuration, the two hosts are setup at the locations which could have longest possible distance.

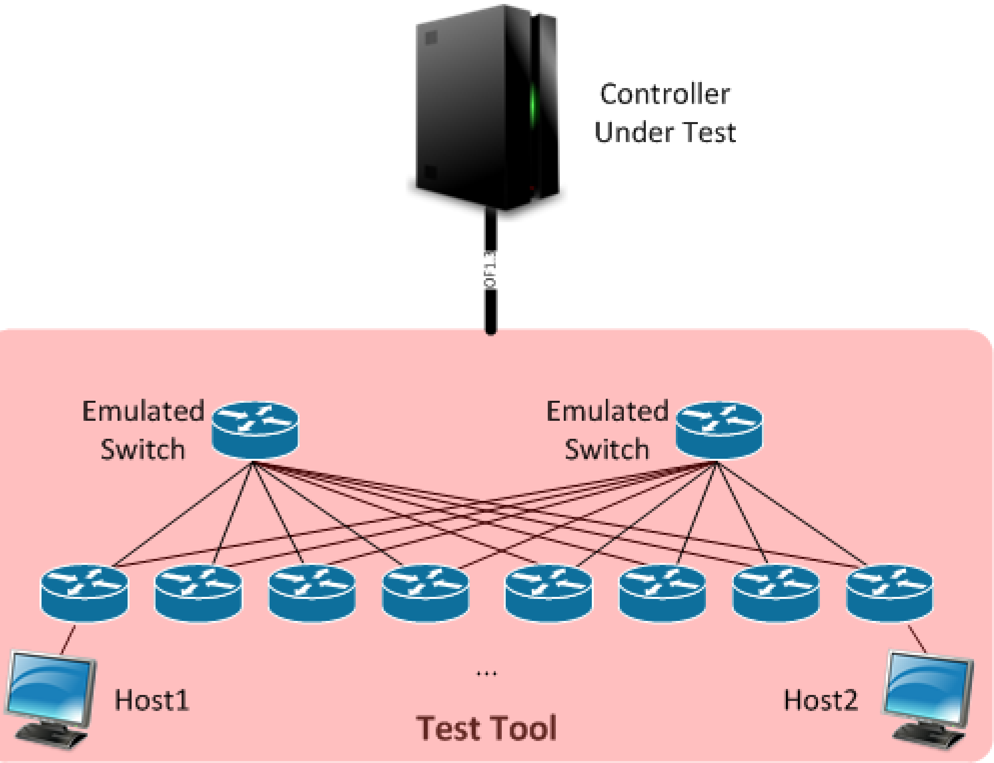


Fig 9.An Example for Reactive Path Failure Recovery Time test

Methodology

1. Start one controller running in standalone mode
2. Start capture
3. Switch (es) start to establish connections to the controller.
4. Wait until the topology discovery completed.
5. Controller starts a path failure recovery application
6. Host1 sends a packet to Host2
7. Wait until controller setups only one end-to-end path between Host1 and Host2
8. Host1 keeps sending packets to Host2
9. Bring down a link in the current traffic path
10. Record the time T1 when controller receives the topology change event message
11. Record the time T2 when the Host2 receives the first packet after the topology change event message
12. Reactive path failure recovery time T = T2 - T1
13. Iterate this test for different topology types and number of nodes
14. Iterate this test for controller instances running in cluster mode

6. CONCLUSION & FUTURE WORK

In this white paper, a few of SDN controller test methodologies are introduced. They are designed based on the real situation and requirements of network users when deploying their commercialized network. One open-source SDN controller has been selected as the controller under test in order to provide quantitative demonstrations of these tests. The results show that the controller running in standalone mode is stable and functional. Although some of the performance is needed to be further improved, it is able to serve as a reliable controller for a small scale of SDN network already. Regarding the cluster mode, the synchronization mechanism is not mature yet. This may cause overheads or even errors in the behavior of the cluster and eventually lead to network failure in large scale of SDN network. The test methodologies could be replicated by anyone who would like to benchmark the performance of his controller and any feed backs or discussions are welcome.

The performance test cases covered in this white paper are treated as a start point of this work. For each of the test cases, more specific configurations could be added and more new test cases will be introduced to this work. Currently, the southbound interface is the focus and only one particular version of OpenFlow protocol is mentioned. In the future, the northbound interface will also be taken into account. The test lab designs the test cases mainly depending on the needs of network users. By providing the users with the data and results they care most, it is an effective way to move forward the commercialization of SDN technology and benefit the whole eco-system.

Now the test lab is seeking for more collaboration with the test tool vendors and controller vendors in order to demonstrate the performance of their products better. It will be perfect if the test tools could emulate more scenarios and the controller could provide APIs to equip the tester with the ability for more precise status monitoring. Also in international standard organizations, the work of standardizing the controller performance test methodologies is also being performed.

Besides performance test, the security of SDN controller is also a main concern for SDN network users. In the future, test cases regarding security will be designed and implemented; methodologies and test results will be included in dedicated test reports. Some security test cases are quite similar to performance test cases such as the test of denial of service. More specific security test cases regarding the OpenFlow protocol and northbound interface will also be covered. Combine the results of both performance test and security test, network users could obtain a more comprehensive view regarding their deployment of SDN network.

7. GLOBAL SDN CERTIFIED TESTING CENTER (SDNCTC)

The global SDN certified testing center (www.sdnctc.com) is a neutral third party SDN/NFV testing and certificating lab, devoting itself to R&D, testing and certificating, deployment and promotion of SDN/NFV technology. SDNCTC actively works on the development of test specification, test tools and testing and certificating ecosystem under the collaboration with SDN/NFV international standards organizations like ONF. Meanwhile, SDNCTC also provide a neutral testing and certification services globally, accelerating the improvement of technology and products, guarantee the commercial deployment of SDN/NFV.