UNIVERSITY OF ALBERTA

MINT 709 **Capstone Project Report**

Performance Analysis of SDN Devices and Controllers

Anmol Sachdeva

Supervisor: Mr. Gurpreet Nanda

Performance Analysis of SDN Devices and Controllers

By

Anmol Sachdeva

A Project Submitted in Partial

Fulfillment of the Requirements for

the Degree of Master of Science in

Internetworking

at

The University of

Alberta

April 2017

Acknowledgements

I would like to express the deepest appreciation to my advisor Dr. Mike MacGregor, for his useful and continuous guidance throughout the graduate program. I would like to thank my supervisor, Mr. Gurpreet Nanda for his aspiring guidance and supervision throughout the project. I would also like to show my gratitude to Mr. Shahnawaz Mir who has always been available for all kinds of help and assistance. Thank you all for your unwavering support.

Table of Contents

Section 1

Section 2

Section 3

Section 4

Section 5

Introduction

Despite SDN's continuing growth in popularity, very few studies have taken place that deal with performance evaluation of SDN architectures. A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood (On Controller Performance in Software-Defined Networks) did a research in which they focused mainly on performance evaluation of the control plane of SDN. They covered a limited number of controllers-NOX-MT, Beacon and Maestro [1][2].

"An Open Framework for OpenFlow Switch Evaluation" was conducted by C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. W. Moore in which they proposed a tool for evaluating performance of one specific SDN architecture i.e. several OpenFlow implementations. Raw performance of OpenFlow without comparison to other SDN solutions, or to non-SDN network systems was measured in this study [1].

Another study of data plane performance evaluation of OpenFlow softswitch (OpenFlowSwitching: Data Plane Performance) was carried out by A. Bianco, R. Birke, L. Giraudo, and M. Palacin using different frame sizes and rules. The results showed that OpenFlow and a more complicated native (non-SDN) application of routing outperform a simpler native switching application [1][2].

M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia on "Modeling and performance evaluation of an OpenFlow architecture" evaluated the forwarding speed of OpenFlow. In this study, focus was on the effects the controller has on the forwarding plane [1].

All the researches described above does not give the idea about which SDN controllers would perform best under different switch loads. Neither they discuss about the effect on performance under different circumstances. In this project, I have taken two industry's leading open source SDN controllers and compared them against each other under different switch loads to find out which one among the two turns out to be a winner.

The Need for Performance Analysis of SDN Controllers

Over the last few years there has been a significant growth that IT industry has shown in the area of Software Defined Networking. Given that SDN controllers are a new class of product, many IT companies have a hard time in deciding which SDN controller would meet the company's requirements and would be best for them.

By doing the performance analysis of SDN controllers, it would become easier for the IT organizations to decide and choose the best SDN controller according to the organizations' needs. The performance evaluation of the controllers would give organizations an idea that how the controllers are going to perform in their network infrastructure and based on that evaluation, they could decide and finalize which SDN controller to opt for among the shortlisted controllers.

Project Purpose

The major purpose of this project is to do the performance benchmarking of SDN controllers and compare them. There are numerous open-source SDN controllers available in the market today like Floodlight Open SDN Controller, OpenDaylight SDN Controller, OpenContrail SDN Controller, FlowVisor OpenFlow Controller, Ryu OpenFlow Controller, ONOS SDN Controller, NOX, POX, Beacon and the list goes on. However, the SDN controllers that I have chosen for the comparison are Floodlight and ONOS.

One of the main reasons for choosing these two controllers is that both are one of the most prominent figures and major contenders in the industry. The other reasons include both the controllers are written in Java, both are open source and are carrier-grade oriented as well.

The tool that has been used to do the performance benchmarking of the SDN controllers is PktBlaster SDN Testing Tool from Veryx.

About Veryx PktBlaster SDN Testing Tool

Veryx PktBlaster SDN tool is a performance benchmarking and network emulation distinct solution for SDN controllers. It is a software based solution that can run on any Linux based X86 server. The tool offers accurate measurements of throughput, latency and robustness parameters. PktBlaster can test the controllers with multiple network types and the GUI allows configuration of multiple switch groups for the same. Both the OpenFlow versions can be tested simultaneously by configuring the networks. The test methodologies used for benchmarking is aligned with the IETF benchmarking draft which Veryx has coauthored [3].

The tool has a browser based GUI application and also supports an API based usage for integration with automated test frameworks. The entry can be created for the SDN controller that has to be tested. The controller to switch connections is established using TCP. The tool offers a traffic profile tab which enables traffic types for testing. MAC address, packet size, IP address, port numbers and number of flows can be configured in the traffic profile. The tool supports flooded and custom traffic types including TCP, UDP, ARP and custom.

PktBlaster tool offers the flexibility to configure the verification parameter as flow-mod or packet-out for each tests. The tool supports OpenFlow as their southbound API and also provides rest API for the northbound interfaces.

The feature that makes this tool unique is that it can be executed in the CLI mode unlike other testing solutions which gives testers the flexibility to run regression and automated tests by reducing the test cycle duration. The tool supports up to 1000 switches for the performance measurement. Each configured switch has the ability to send more than a million packet-ins towards the controller based upon the count selected in the traffic profile. Using this tool, the tester can determine throughput, latency and robustness [3].

A throughput test determines the rate at which the controller configures the flow in the switches. The latency test gives exact time in milliseconds which the controller takes to configure a flow in the switch. Both these tests can be performed in Normal mode and Incremental Switch mode.

A Normal mode test allows the switches to connect to the controller simultaneously. The Incremental Switch mode determines the scalability of the controller by increasing the number of switches by 25% for each test iteration which subjects the controller to variable load testing

and determines that where the controller performance is highest and also the consistency of the performance. The tool has the flexibility to change number of test iterations and the test duration for both the modes.

While performing a test, the GUI shows the progress of the test and a live graph is generated reflecting the values captured for the controller. Detailed statistics for the total number of transmissions and receptions for every switch are provided which gives the tester granular details of the controller [3].

The robustness tests help to understand how controller handles enormous packets. Unlike other tools, PktBlaster offers an added advantage of determining controller behavior during network impairment scenarios. The packets can be malformed. The percentage of erroneous traffic are configurable. While a packet-in message is sent, the message type is corrupted to an invalid value. The erroneous packets are introduced incrementally in the test and a live graph is generated showing the robustness values.

For Throughput Tests and Latency Tests the maximum number of switches that can be configured in this tool is 1000 and for Robustness Tests the maximum number of switches is 64.

This tool can be very useful for the equipment vendors while building the controllers and for the service providers to understand and compare different controller behavior in their networks. It serves as an ideal solution for testing SDN controllers [4].

About Open Network Operating System (ONOS)

The ONOS (Open Network Operating System) project is an open source community hosted by the Linux Foundation. The goal of the project is to create a software-defined networking (SDN) operating system for communications service providers that is designed for scalability, high performance and high availability [5].

ONOS is architected to provide high availability, scalability, performance and rich northbound and southbound abstractions.

ONOS is developed by ON.Lab in partnership with a community of leading service providers, vendors and researchers. Founding members who are funding and contributing to the ONOS initiative include AT&T, NTT Communications, Ciena, Fujitsu, Huawei, Intel, NEC; and members who are collaborating and contributing to ONOS include ONF, Infoblox, SRI, Black Duck Software, Internet2, GARR, CNIT and Create-Net.collaborative projects [6].

The Open Networking Lab (ON.Lab) along with other industry partners including AT&T and NTT Communications released the ONOS source code On December 5, 2014 to start the open source community. On October 14, 2015, the Linux Foundation announced that ONOS had joined the organization as one of its collaborative projects.

ONOS is written in Java and is designed to operate as a cluster of nodes that are identical in terms of their software stack and can resist failure of individual nodes without affecting its ability to control the operation of network. One big advantage of ONOS is that its system architecture is not directly tied to standard protocols and models. It has its own set of high-level abstractions and models. The platform provides applications with a number of high-level abstractions, through which the applications can learn about the state of the network and through which they can control the flow of traffic through the network [5].

The first ONOS release was Avocet on December 5, 2014 Followed by Blackbird (February 28, 2015), Cardinal (May 31, 2015), Drake (September 18, 2015), Emu (December 18, 2015), Falcon (March 10, 2016), Goldeneye (June 24, 2016), Hummingbird (Sept. 23, 2016) and Ibis (Jan 12, 2017).

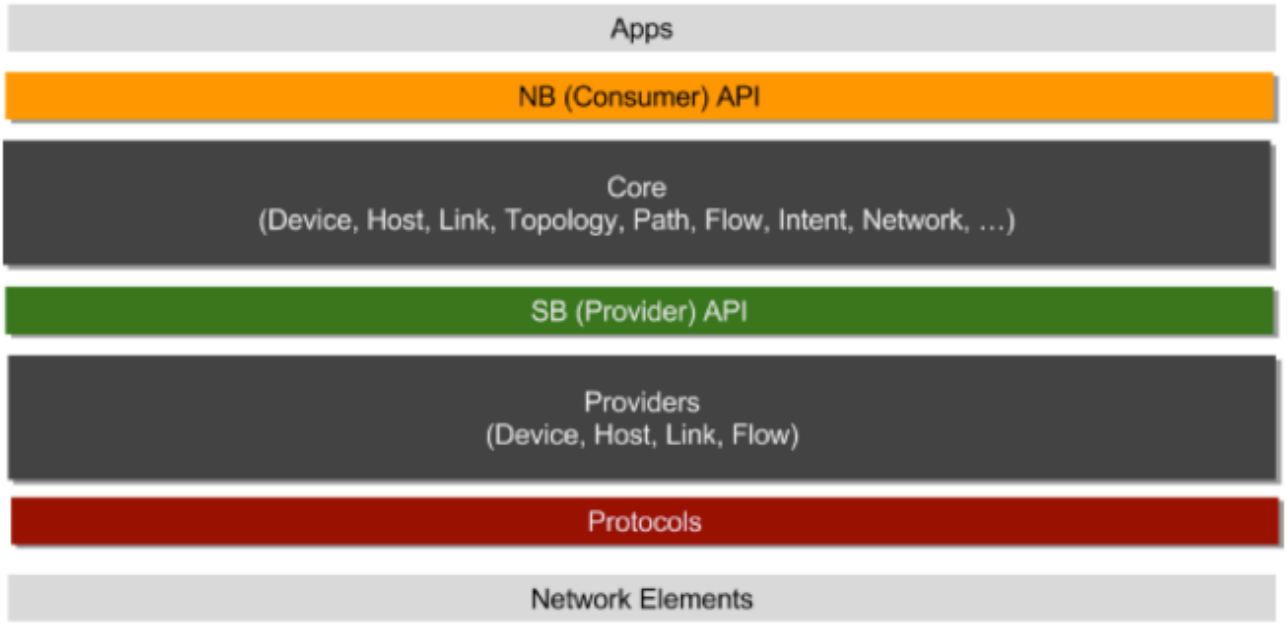


Fig. 1: ONOS System Tiers [7]

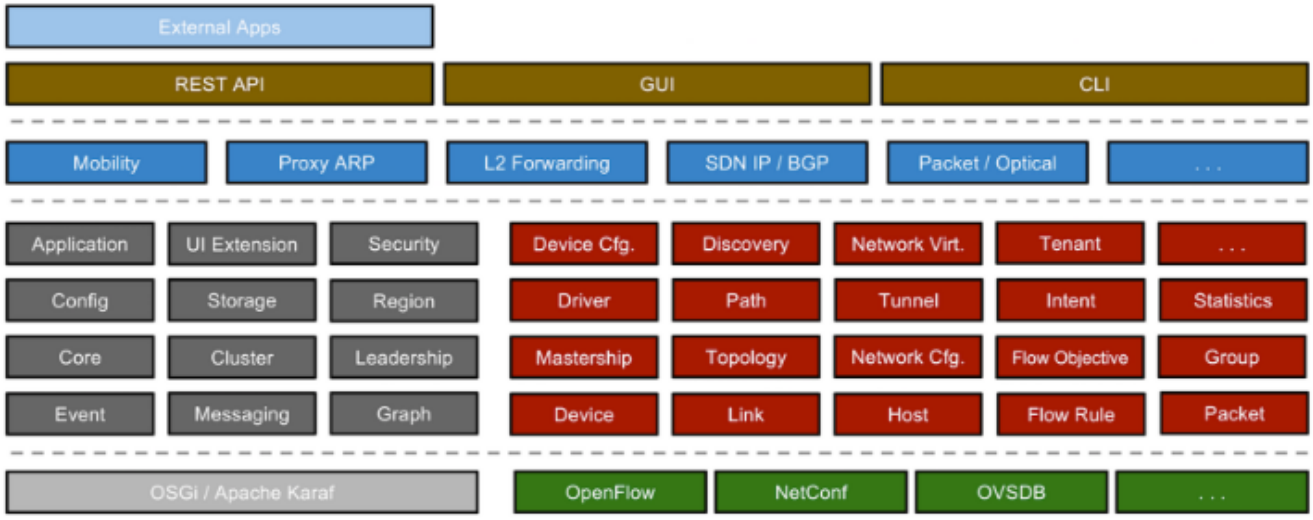


Fig. 2: ONOS Subsystems [7]

The above figure illustrates the various subsystems that are part of ONOS today and a few that are planned in the near future.

About Floodlight

The Floodlight Open SDN Controller is an enterprise-class, Apache-licensed, Java-based OpenFlow Controller. It is supported by a community of developers including a number of engineers from Big Switch Networks.

Floodlight is designed to work with the growing number of switches, routers, virtual switches, and access points that support the OpenFlow standard.

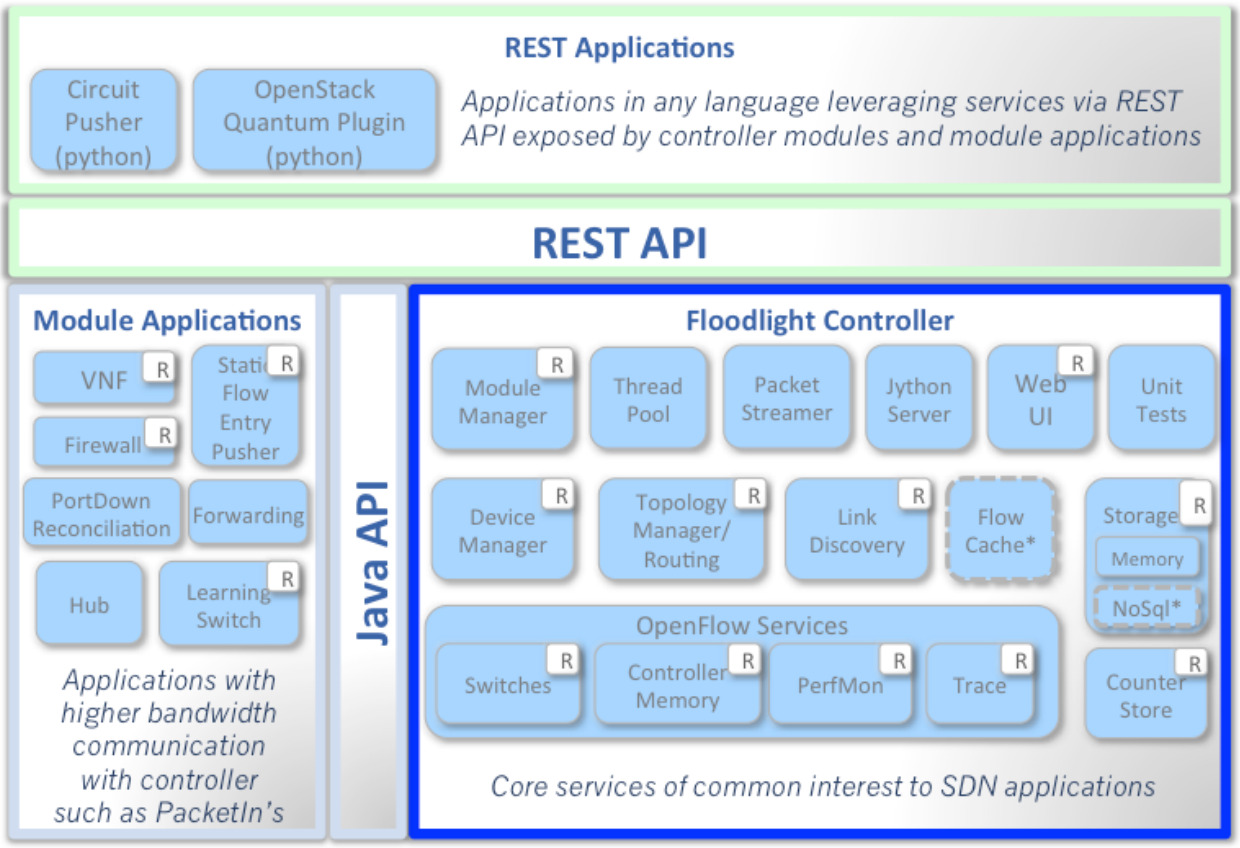


Fig. 3: Floodlight Architecture [8]

Features-

-Offers a module loading system that make it simple to extend and enhance.

-Easy to set up with minimal amount of dependencies.

-Support for OpenStack (link) cloud orchestration platform

-Handles mixed OpenFlow and non-OpenFlow networks

-Supports a wide range of virtual and physical OpenFlow switches

-Designed to be high-performance. The controller is multithreaded from the ground up

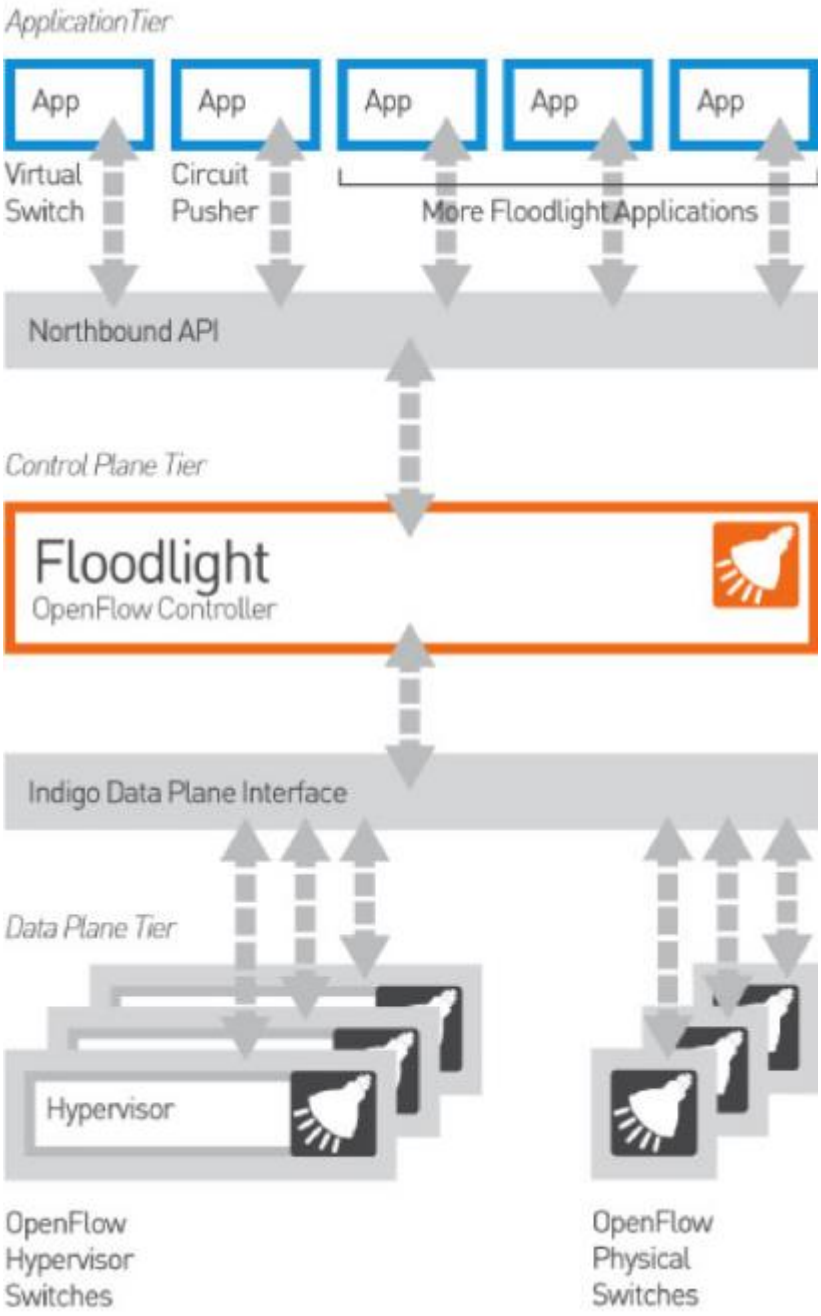


Fig. 4: How Floodlight Controller works in SDN Environments [9]

The different versions of Floodlight include Version 0.85, Version 0.90, Version .91, Version 1.0, Version 1.1, Version 1.2 and Nightly.

Test Setup

Both ONOS and Floodlight SDN controllers have been built over different Ubuntu 14.04 (Trusty Tahr) 64-bit LTS versions with the following configurations-

Virtual Disk Size:	50 GB
Memory (RAM):	16 GB
CPUs:	8

Details of the SDN Controllers

1. Open Network Operating System (ONOS)

Software Build Version:	1.5 (Falcon)
IP Address:	10.3.34.168
OpenFlow Port:	6633
OpenFlow Version:	1.0
Mode of Connection:	TCP

2. Floodlight

Software Build Version:	Version 1.2
IP Address:	10.3.34.165
OpenFlow Port:	6653
OpenFlow Version:	1.0
Mode of Connection:	TCP

Test Configuration for Throughput Tests

Test Type:	Throughput
Test Mode:	Normal & Incremental Switch
Flow Measurement by:	Flow-mod
Test Duration (sec):	1
Number of Iterations:	4
Iteration Delay (sec):	1
Warmup Time (sec):	1

Switch Configuration for Throughput Tests

OpenFlow Version:	1.0
Start MAC Address:	00:00:00:00:00:01
Traffic Profile:	TCP
Packet Length:	80
# of Group Tables:	1
Max. Entries per Flow Table:	65,536
Switch Capabilities:	Flow Stats, Table Stats, Port Stats
# of Ports:	8
# of Flows:	2
# of Flow Tables:	1
# of Meter Tables:	1
Switch Buffers Size:	0

Test Configuration for Latency Tests

Test Type:	Latency
Test Mode:	Normal & Incremental Switch
Flow Measurement by:	Flow-mod
Test Duration (sec):	1
Number of Iterations:	4
Iteration Delay (sec):	1
Warmup Time (sec):	1

Switch Configuration for Latency Tests

OpenFlow Version:	1.0
Start MAC Address:	00:00:00:00:00:01
Traffic Profile:	TCP
Packet Length:	80
# of Group Tables:	1
Max. Entries per Flow Table:	65,536
Switch Capabilities:	Flow Stats, Table Stats, Port Stats
# of Ports:	8
# of Flows:	2
# of Flow Tables:	1
# of Meter Tables:	1
Switch Buffers Size:	0

Test Configuration for Robustness Tests

Test Type:	Robustness
Flow Measurement by:	Flow-mod
Test Duration (sec):	1
Number of Iterations:	4
Malform Field:	OF Version
OF Version:	99
Malformed Packets (%):	1
Iteration Delay (sec):	1
Warmup Time (sec):	1

Switch Configuration for Robustness Tests

OpenFlow Version:	1.0
Start MAC Address:	00:00:00:00:00:01
Traffic Profile:	TCP
Packet Length:	80
# of Group Tables:	1
Max. Entries per Flow Table:	65,536
Switch Capabilities:	Flow Stats, Table Stats, Port Stats
# of Ports:	8
# of Flows:	2
# of Flow Tables:	1
# of Meter Tables:	1
Switch Buffers Size:	0

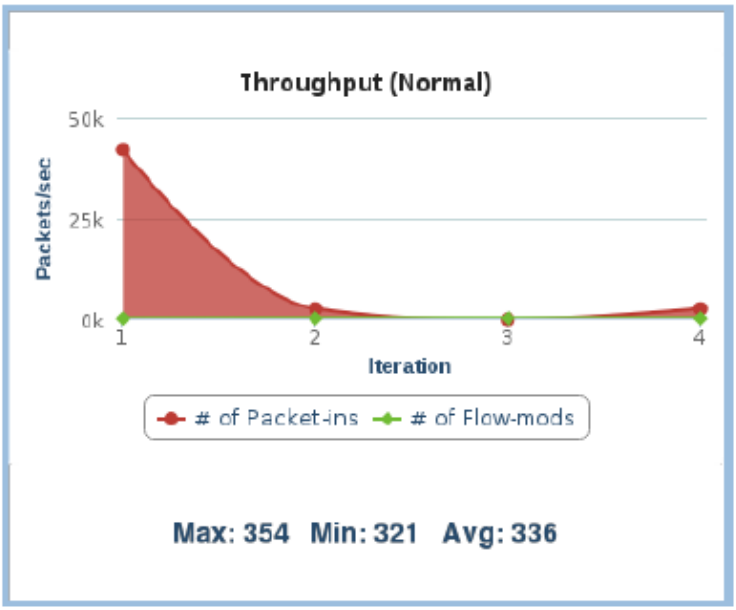
Performance Analysis of Throughput Tests in Normal Mode

In the throughput test, the controller is loaded to the maximum with Packet-in messages and Throughput is measured on the basis of Flow-mod messages that are sent back by the controller in response to Packet-in messages.

The Throughput performance test in Normal Mode was carried out for ONOS and Floodlight controller with 4, 8, 16, 32, 64 and 128 switches and ONOS clearly outperformed Floodlight in this test for all the switches.

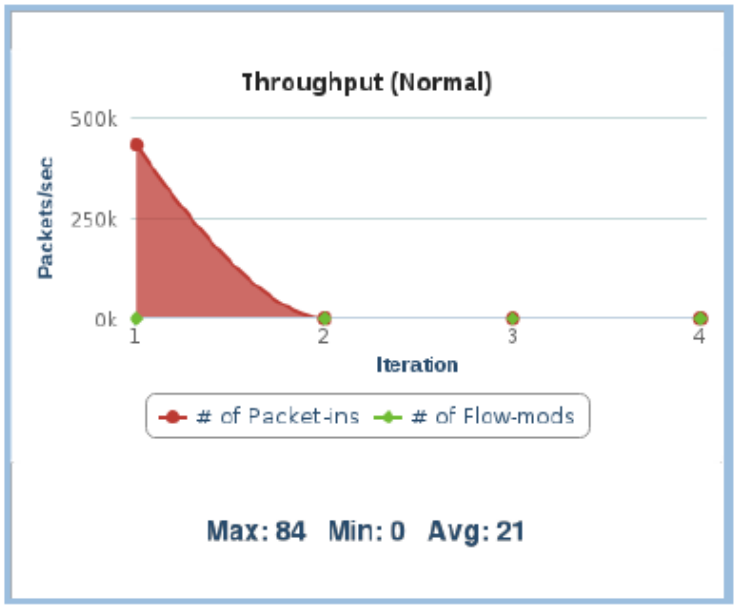
Screenshots of the graphs and comparison details are attached in the following pages.

As we can see, 4 iterations were carried out for the throughput analysis. In response to the Packet-in messages, ONOS generated much higher number of Flow-mod messages as compared to Floodlight in all the 4 iterations for all the tests performed with different number of switches.



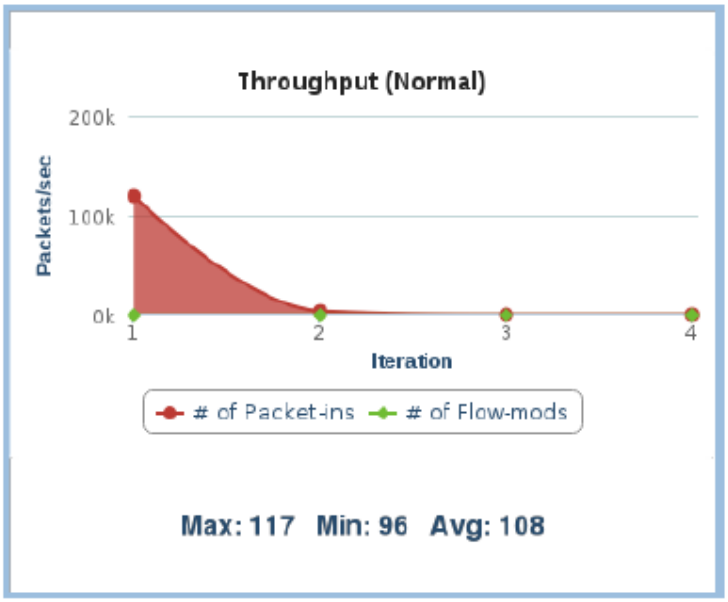
S.No	Iteration	# of Packet-ins	# of Flow-mods
1	1	42,084	321
2	2	2,672	332
3	3	0	340
4	4	2,672	354

Fig. 5: ONOS: 4 Switches (Normal)



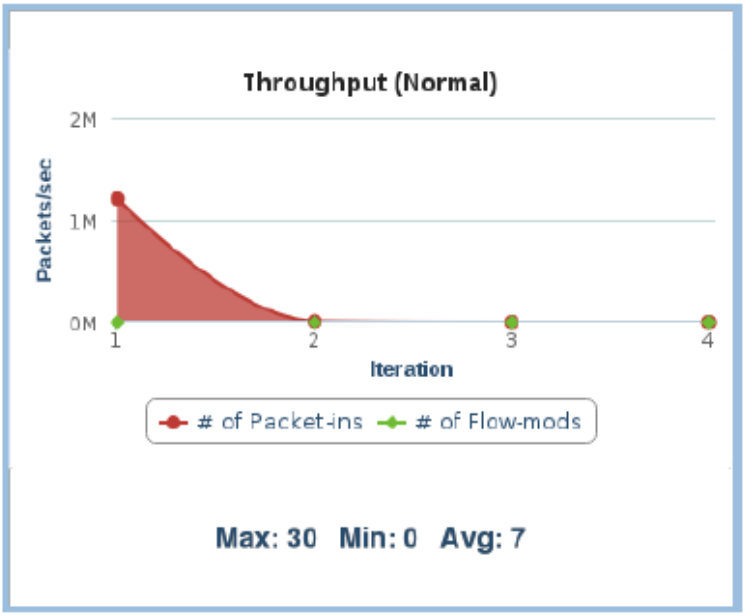
S.No	Iteration	# of Packet-ins	# of Flow-mods
1	1	430,860	84
2	2	0	0
3	3	0	0
4	4	0	0

Fig. 6: Floodlight: 4 Switches (Normal)



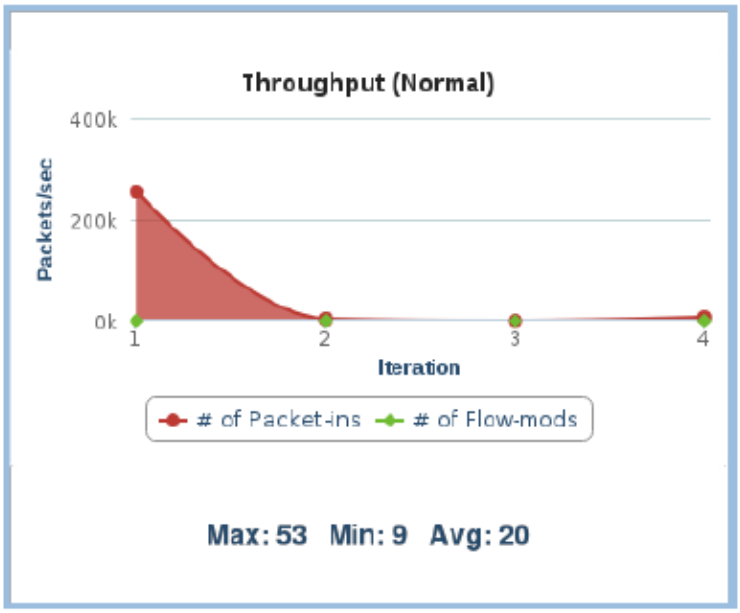
S.No	Iteration	# of Packet-ins	# of Flow-mods
1	1	119,572	113
2	2	4,008	107
3	3	668	117
4	4	668	96

Fig. 7: ONOS: 8 Switches (Normal)



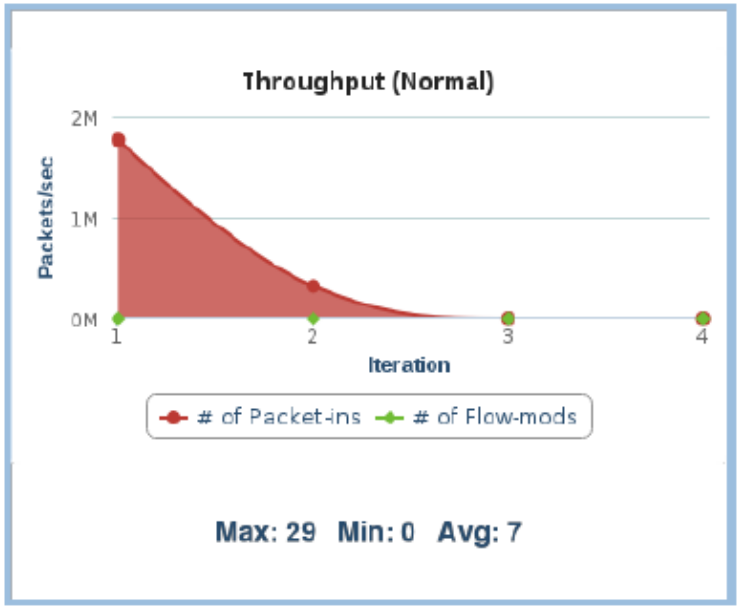
S.No	Iteration	# of Packet-ins	# of Flow-mods
1	1	1,209,080	30
2	2	8,016	0
3	3	668	0
4	4	0	0

Fig. 8: Floodlight: 8 Switches (Normal)



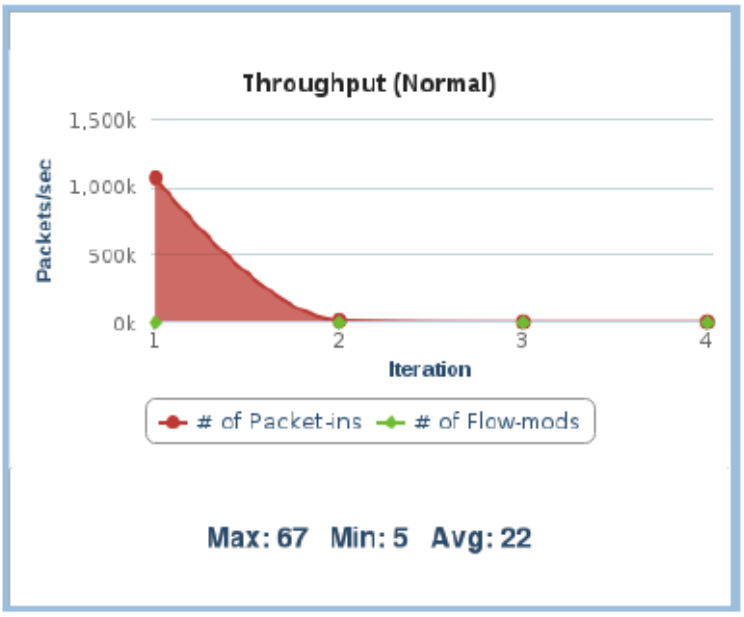
S.No	Iteration	# of Packet-ins	# of Flow-mods
1	1	255,176	53
2	2	3,340	10
3	3	0	9
4	4	6,680	11

Fig. 9: NOS: 16 Switches (Normal)



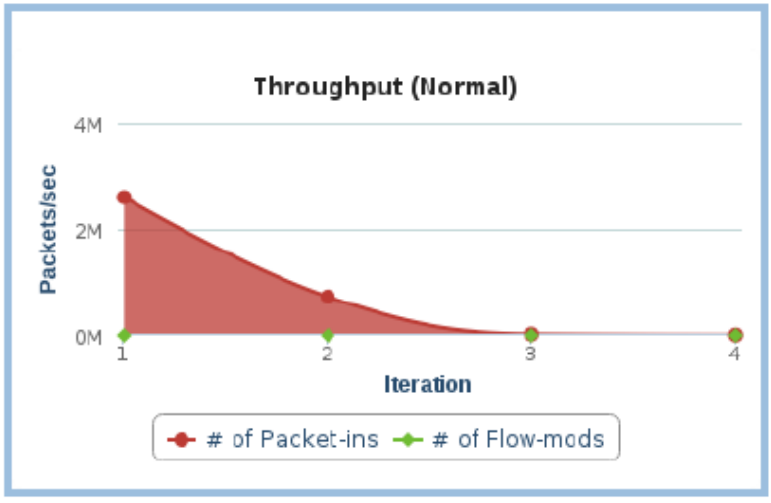
S.No	Iteration	# of Packet-ins	# of Flow-mods
1	1	1,770,200	29
2	2	321,308	0
3	3	668	0
4	4	0	0

Fig. 10: Floodlight: 16 Switches (Normal)



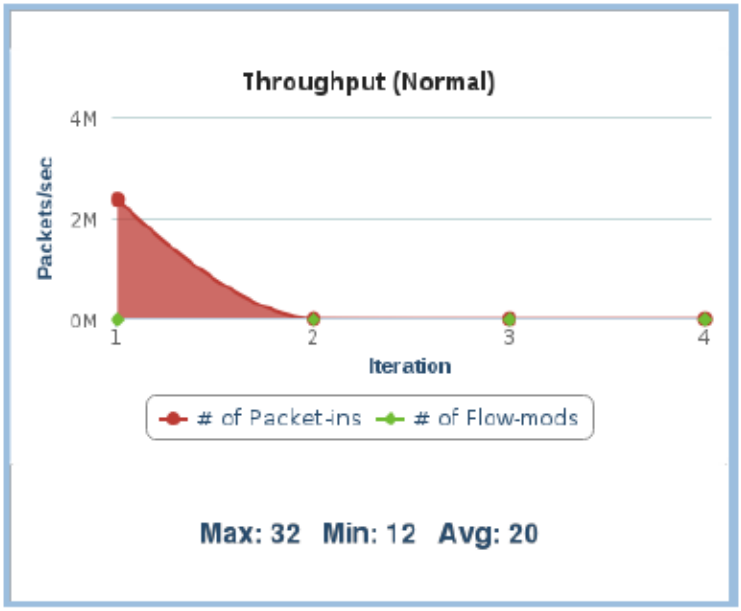
S.No	Iteration	# of Packet-ins	# of Flow-mods
1	1	1,063,456	67
2	2	14,028	8
3	3	4,008	8
4	4	4,008	5

Fig. 11: ONOS: 32 Switches (Normal)



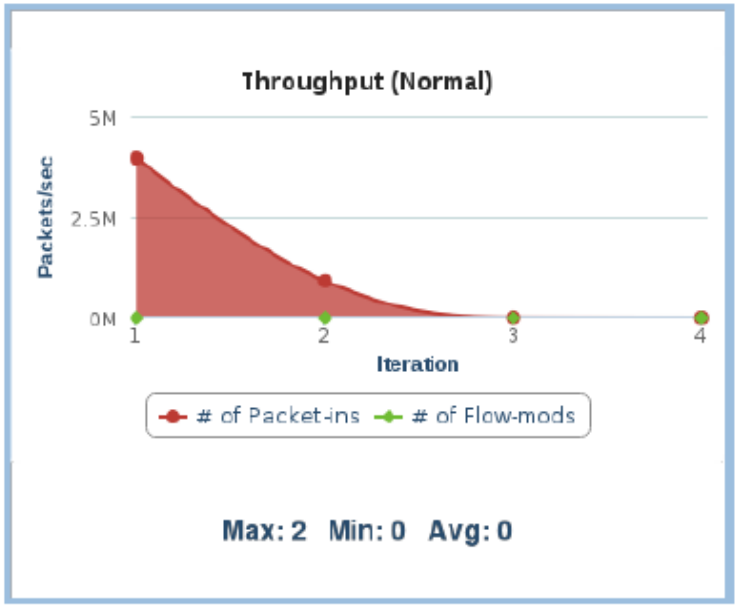
S.No	Iteration	# of Packet-ins	# of Flow-mods
1	1	2,599,188	0
2	2	728,788	0
3	3	21,376	0
4	4	8,016	0

Fig. 12: Floodlight: 32 Switches (Normal)



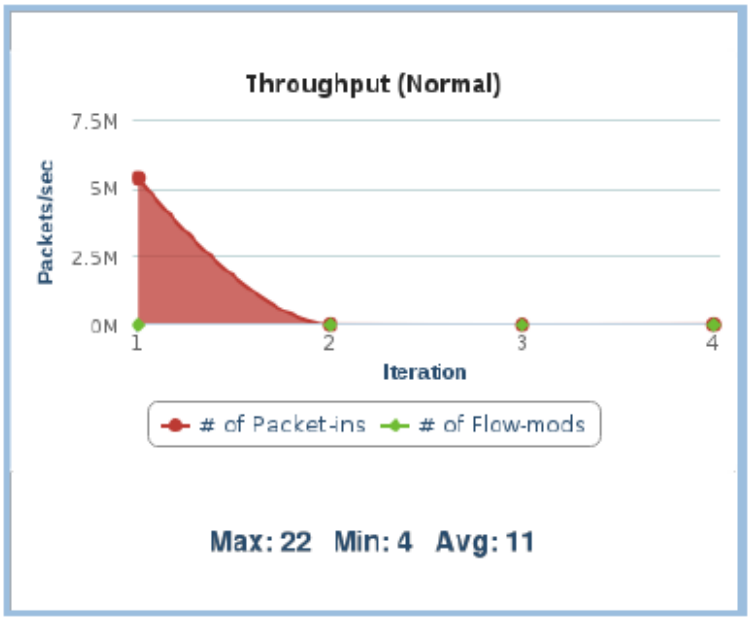
S.No	Iteration	# of Packet-ins	# of Flow-mods
1	1	2,372,068	32
2	2	12,692	12
3	3	12,024	21
4	4	10,688	16

Fig. 13: ONOS: 64 Switches (Normal)



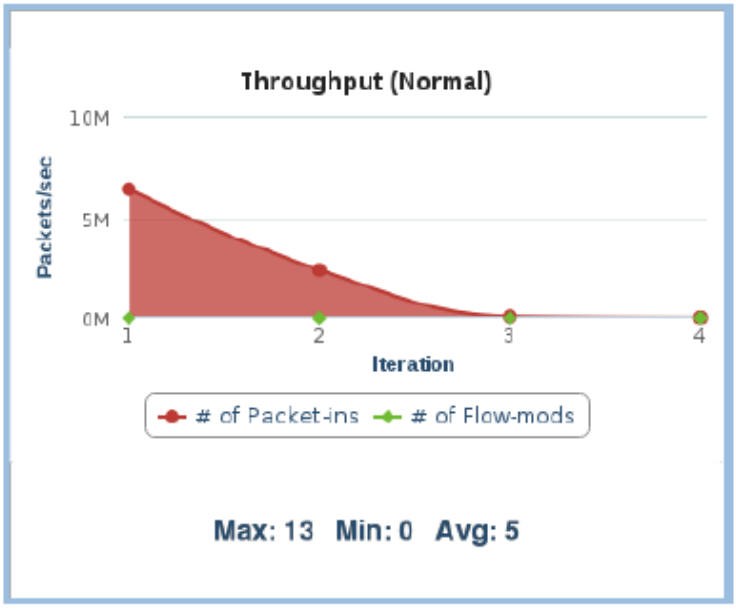
S.No	Iteration	# of Packet-ins	# of Flow-mods
1	1	3,960,572	2
2	2	915,160	0
3	3	6,012	0
4	4	0	0

Fig. 14: Floodlight: 64 Switches (Normal)



S.No	Iteration	# of Packet-ins	# of Flow-mods
1	1	5,352,684	22
2	2	8,016	10
3	3	2,004	8
4	4	3,340	4

Fig. 15: ONOS: 128 Switches (Normal)



S.No	Iteration	# of Packet-ins	# of Flow-mods
1	1	6,392,760	8
2	2	2,383,424	0
3	3	65,464	13
4	4	4,676	0

Fig. 16: Floodlight: 128 Switches (Normal)

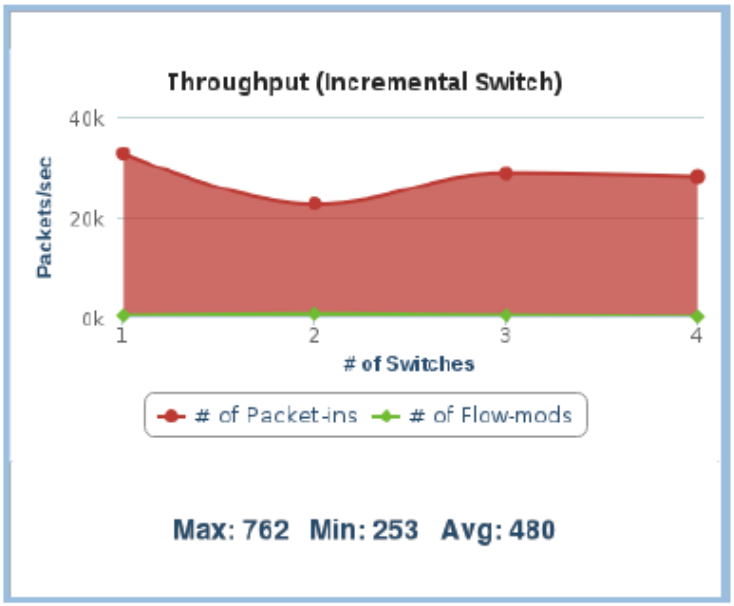
Performance Analysis of Throughput Tests in Incremental Switch Mode

This test is executed in 4 iterations and 25% of total number of configured switches are added for each iteration until the total number of configured switches are included for the test.

Just like the Throughput performance tests in Normal Mode, the test was carried out in Incremental Switch mode for the two controllers and the results were same. ONOS is the clear winner between the two controllers for all the switches.

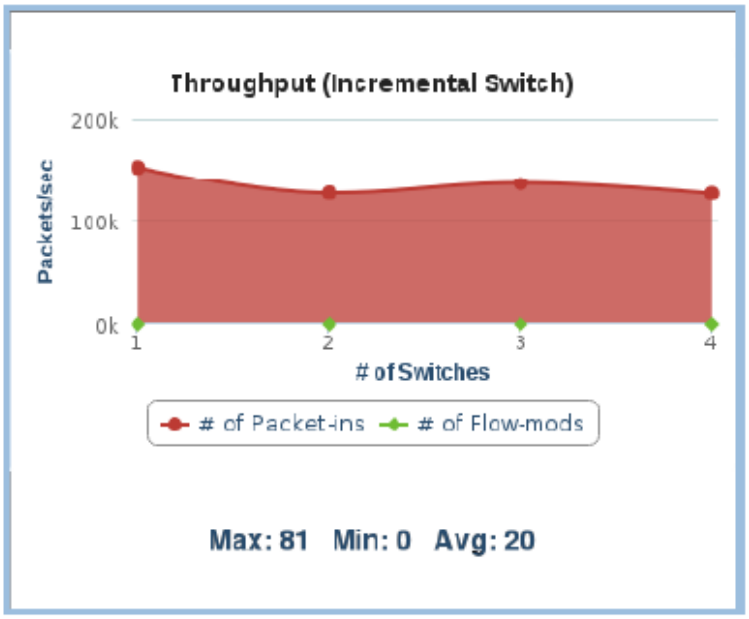
As seen in the screenshots on the following pages, the number of Flow-mod messages generated in response to Packet-in messages are much higher for ONOS as compared to Floodlight in all the iterations for all the tests done with different loads of switches.

For 32 switches, the test stopped after two iterations and for 64 and 128 switches after one iteration for both ONOS and Floodlight due to an internal error of the benchmarking tool.



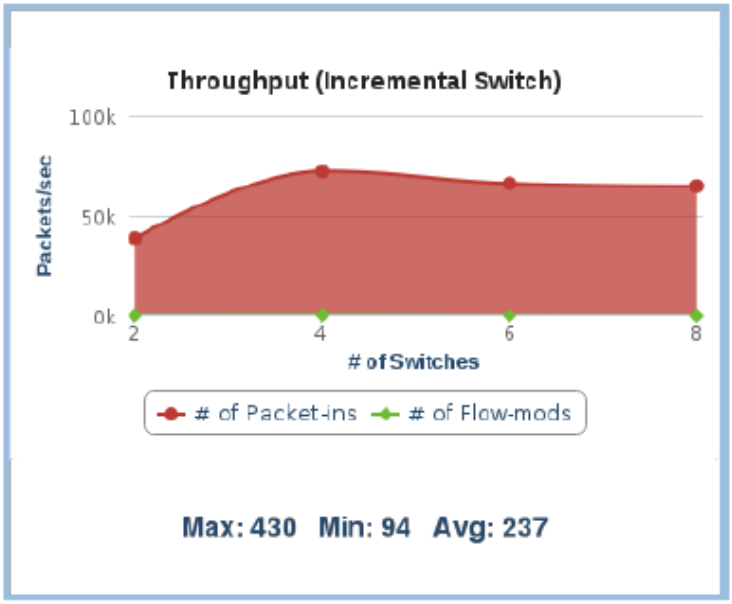
S.No	# of Switches	# of Packet-ins	# of Flow-mods
1	1	32,732	432
2	2	22,712	762
3	3	28,724	476
4	4	28,056	253

Fig. 17: ONOS: 4 Switches (Incremental Switch)



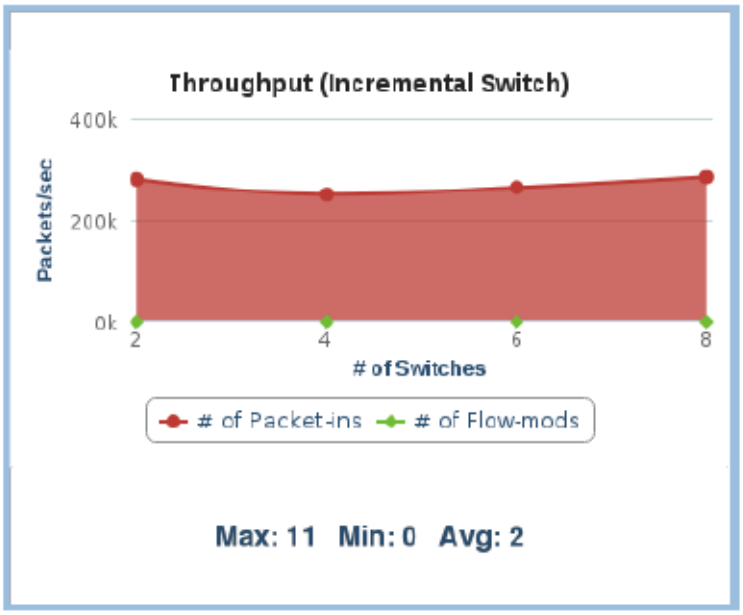
S.No	# of Switches	# of Packet-ins	# of Flow-mods
1	1	152,972	0
2	2	127,588	0
3	3	136,940	81
4	4	126,920	0

Fig. 18: Floodlight: 4 Switches (Incremental Switch)



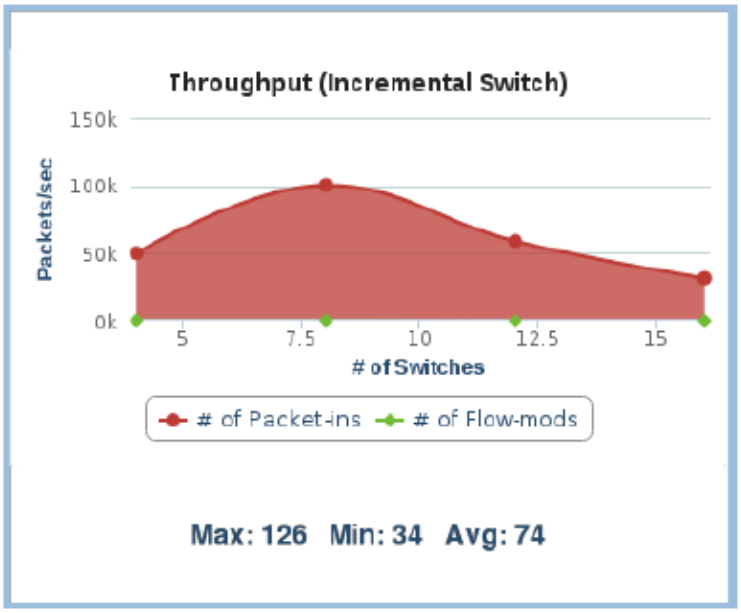
S.No	# of Switches	# of Packet-ins	# of Flow-mods
1	2	38,744	249
2	4	72,144	430
3	6	66,132	176
4	8	64,796	94

Fig. 19: ONOS: 8 Switches (Incremental Switch)



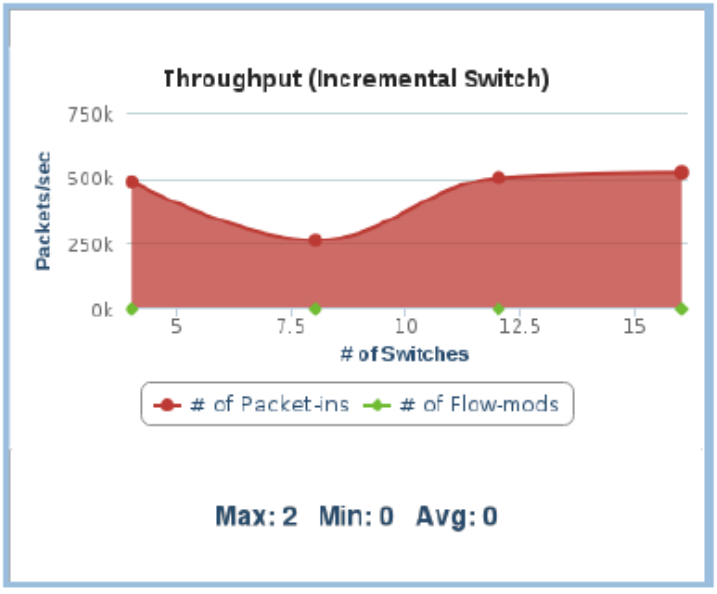
S.No	# of Switches	# of Packet-ins	# of Flow-mods
1	2	279,892	0
2	4	251,168	0
3	6	264,528	11
4	8	284,568	0

Fig. 20: Floodlight: 8 Switches (Incremental Switch)



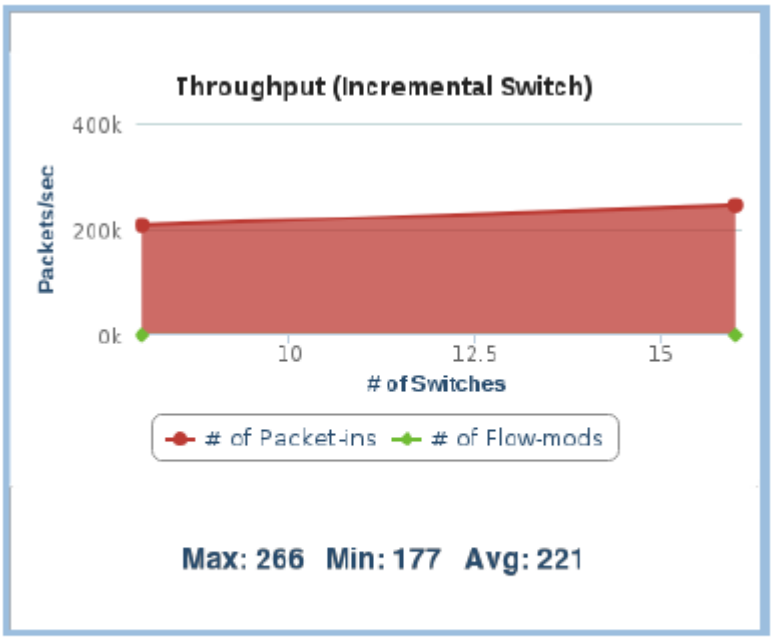
S.No	# of Switches	# of Packet-ins	# of Flow-mods
1	4	50,100	96
2	8	100,200	126
3	12	58,784	40
4	16	31,396	34

Fig. 21: ONOS: 16 Switches (Incremental Switch)



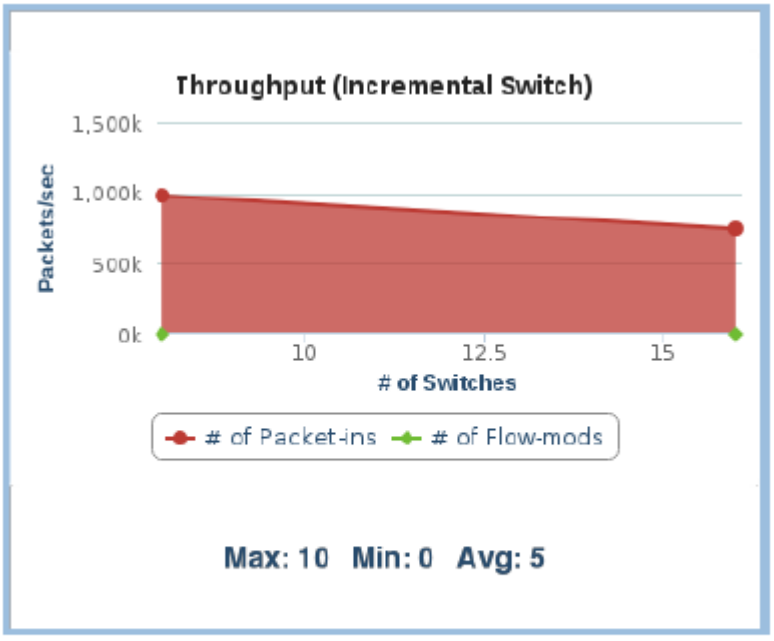
S.No	# of Switches	# of Packet-ins	# of Flow-mods
1	4	485,636	2
2	8	263,192	0
3	12	501,000	0
4	16	521,708	0

Fig. 22: Floodlight: 16 Switches (Incremental Switch)



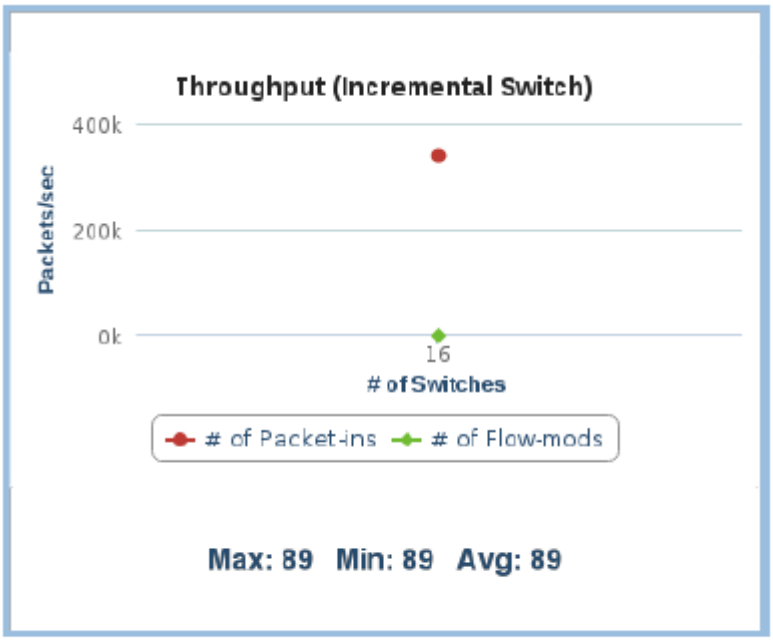
S.No	# of Switches	# of Packet-ins	# of Flow-mods
1	8	207,748	177
2	16	245,156	266

Fig. 23: ONOS: 32 Switches (Incremental Switch)



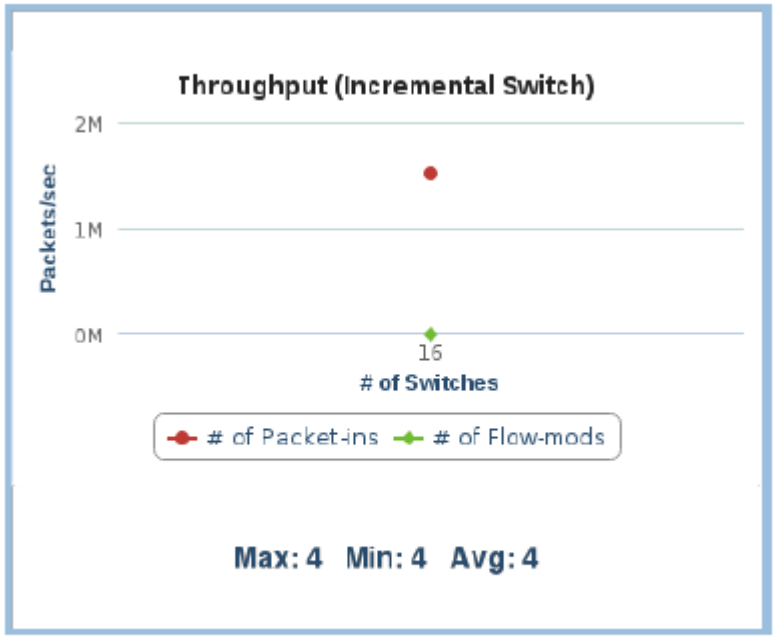
S.No	# of Switches	# of Packet-ins	# of Flow-mods
1	8	981,960	10
2	16	748,160	0

Fig. 24: Floodlight: 32 Switches (Incremental Switch)



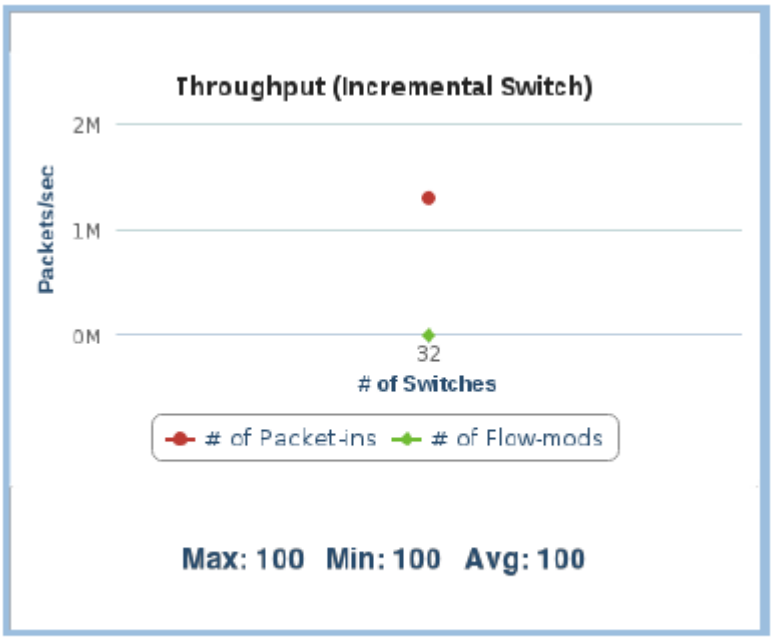
S.No	# of Switches	# of Packet-ins	# of Flow-mods
1	16	339,344	89

Fig. 25: ONOS: 64 Switches (Incremental Switch)



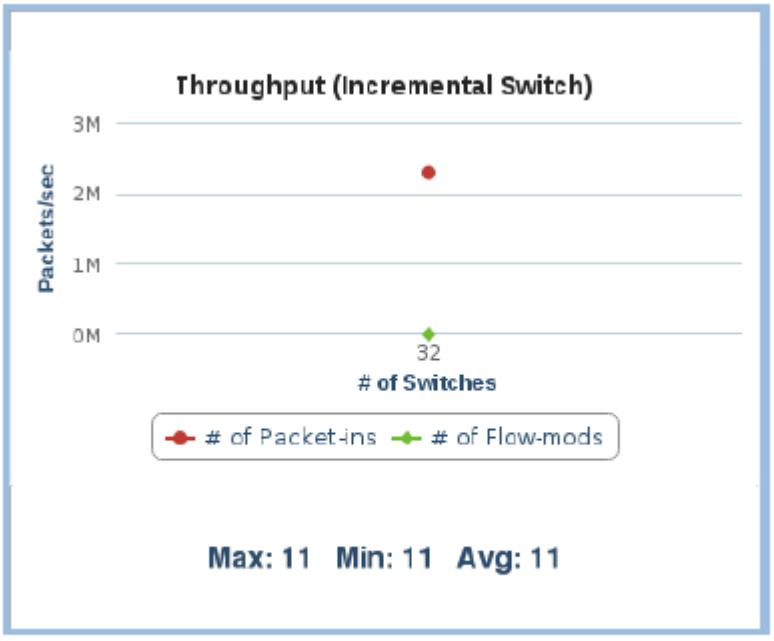
S.No	# of Switches	# of Packet-ins	# of Flow-mods
1	16	1,517,028	4

Fig. 26: Floodlight: 64 Switches (Incremental Switch)



S.No	# of Switches	# of Packet-ins	# of Flow-mods
1	32	1,295,252	100

Fig. 27: ONOS: 128 Switches (Incremental Switch)



S.No	# of Switches	# of Packet-ins	# of Flow-mods
1	32	2,286,564	11

Fig. 28: Floodlight: 128 Switches (Incremental Switch)

Performance Analysis of Latency Tests in Normal Mode

In a Latency test, the tool measures the controller's request processing time under low-load conditions.

Each switch sends a Packet-in message to the controller and then waits for a response (Flow-mod message) from the controller. The difference between the time a packet-in message is sent and a Flow-mode message is received gives the Latency of the controller.

The Latency tests are performed in Normal mode with different number of switches for 4 iterations and the results are quite interesting.

With 4 and 8 switches, for the first iteration, ONOS takes less time as compared to Flooding in responding with Flow-mod messages. However, for the rest 3 iterations, Floodlight emerges as an undisputed winner. It is taking much less time that ONOS in responding with Flow-mod messages.

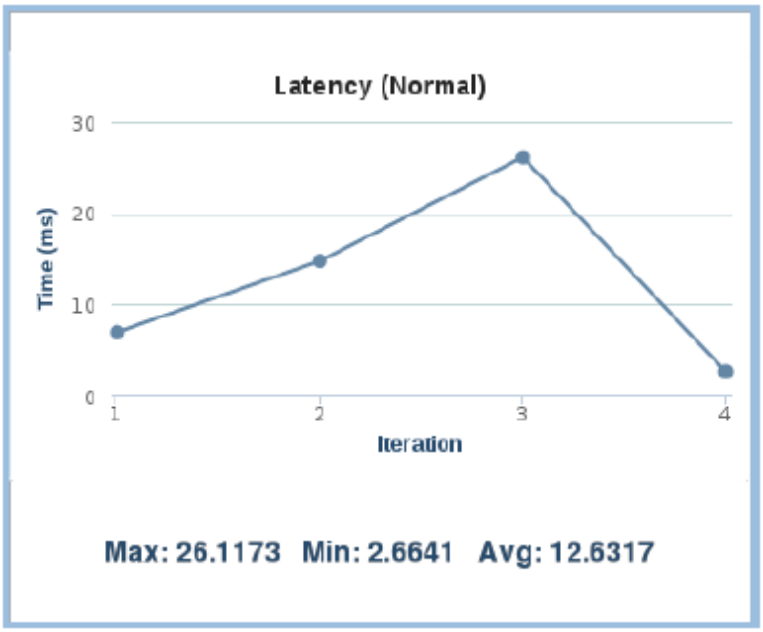
When the test was performed with 16 switches, Floodlight was able to outperform ONOS in 1st, 3rd and 4th iterations. ONOS was able to respond a little quicker than Floodlight only for the 2nd iteration.

With 32 switches, the results were similar as they were with 4 and 8 switches.

With 64 switches, Floodlight again outperformed ONOS for all the 4 iterations.

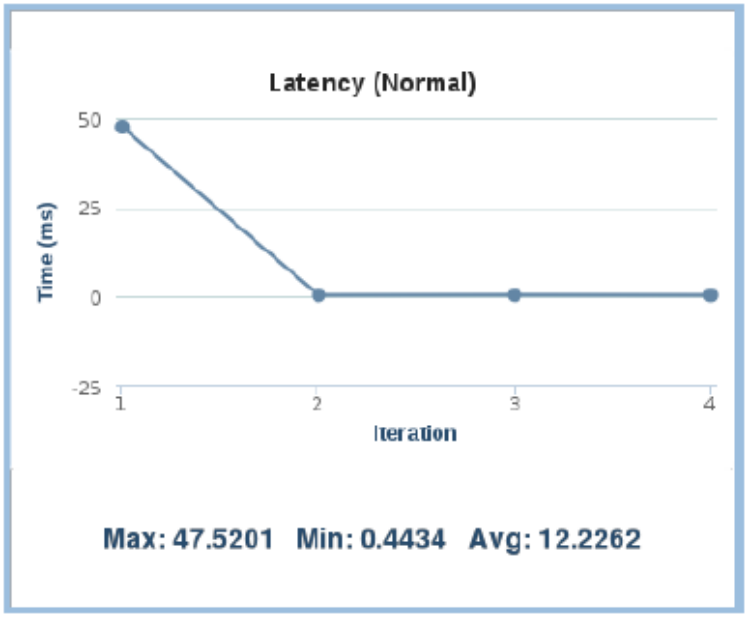
With 128 switches, for the first 2 iterations, ONOS did a better job than Floodlight. However, for the remaining 2 iterations, Floodlight left ONOS behind.

In all these tests, there were lots of variations in the results done with different number of switches. However, talking about the results overall, we would see that Floodlight performed much better that ONOS in the Latency department.



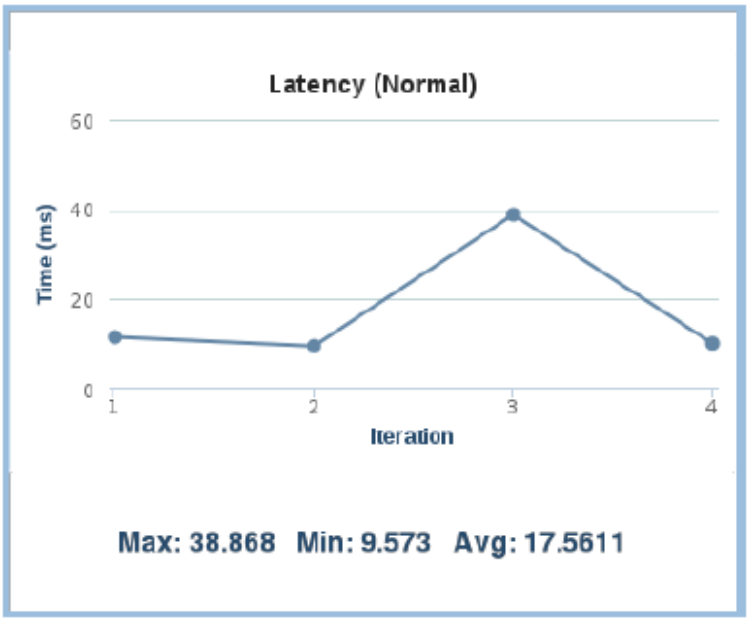
S.No	Iteration	Latency (ms)
1	1	6.9585
2	2	14.7869
3	3	26.1173
4	4	2.6641

Fig. 29: ONOS: 4 Switches (Normal)



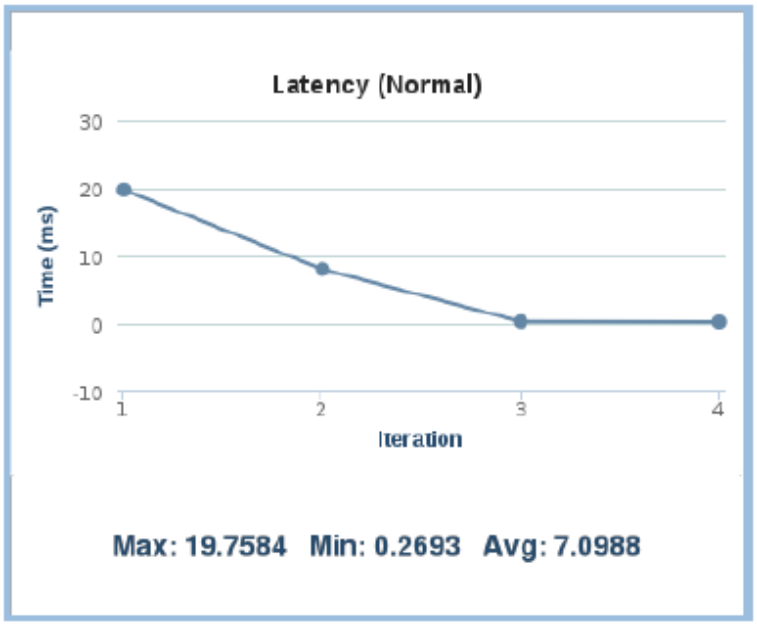
S.No	Iteration	Latency (ms)
1	1	47.5201
2	2	0.4505
3	3	0.4910
4	4	0.4434

Fig. 30: Floodlight: 4 Switches (Normal)



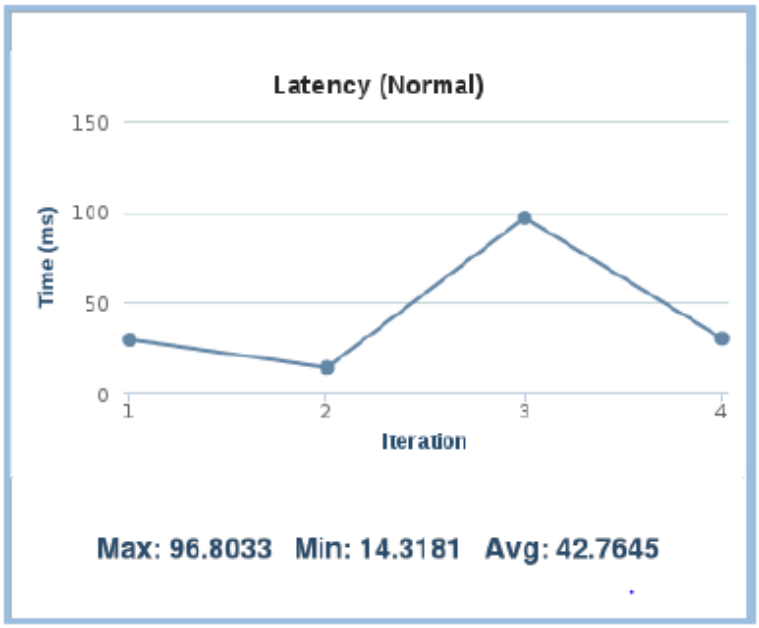
S.No	Iteration	Latency (ms)
1	1	11.6098
2	2	9.5730
3	3	38.8680
4	4	10.1937

Fig. 31: ONOS: 8 Switches (Normal)



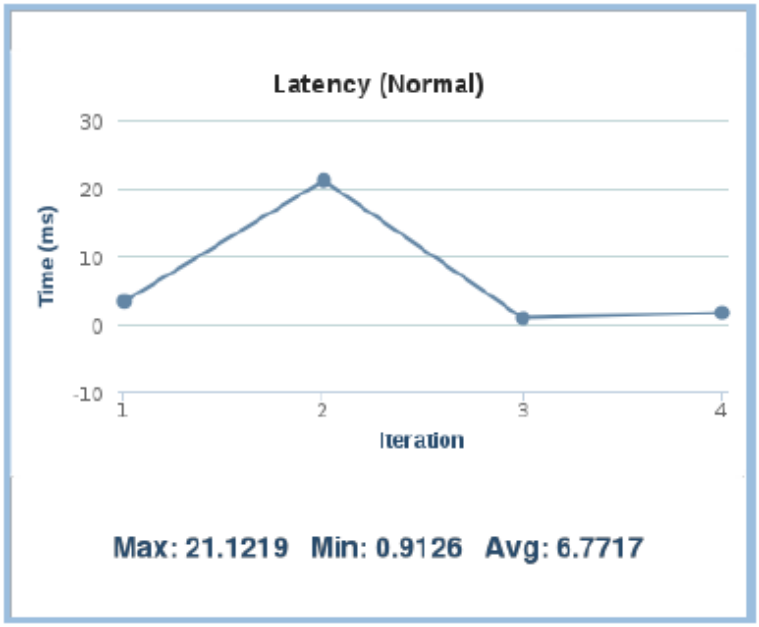
S.No	Iteration	Latency (ms)
1	1	19.7584
2	2	8.0445
3	3	0.3231
4	4	0.2693

Fig. 32: Floodlight: 8 Switches (Normal)



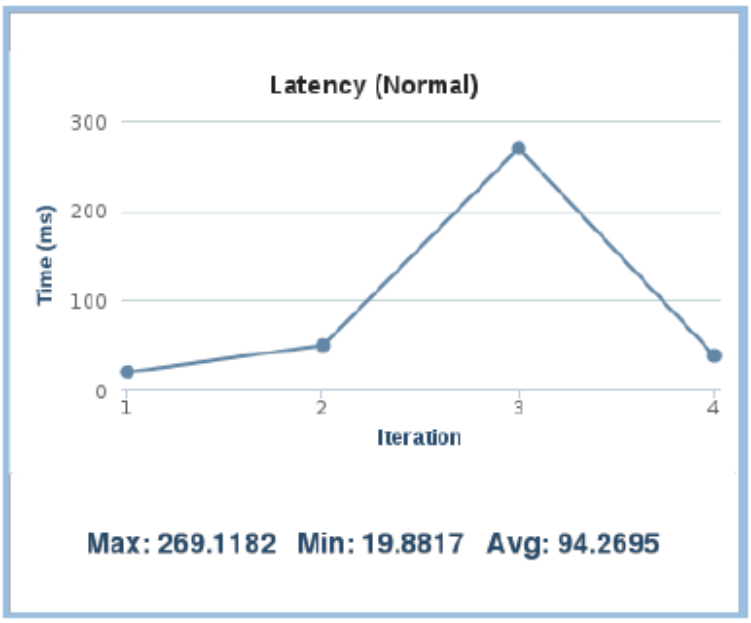
S.No	Iteration	Latency (ms)
1	1	29.6014
2	2	14.3181
3	3	96.8033
4	4	30.3353

Fig. 33: ONOS: 16 Switches (Normal)



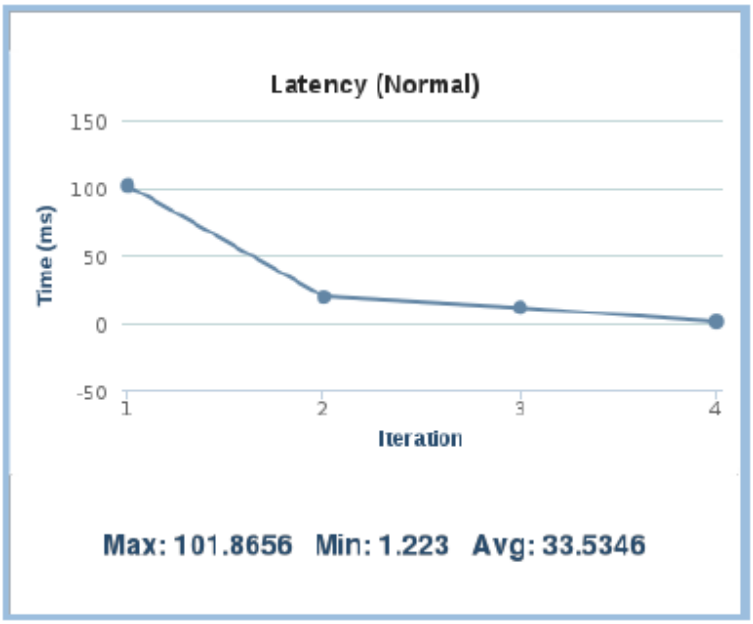
S.No	Iteration	Latency (ms)
1	1	3.3774
2	2	21.1219
3	3	0.9126
4	4	1.6751

Fig. 34: Floodlight: 16 Switches (Normal)



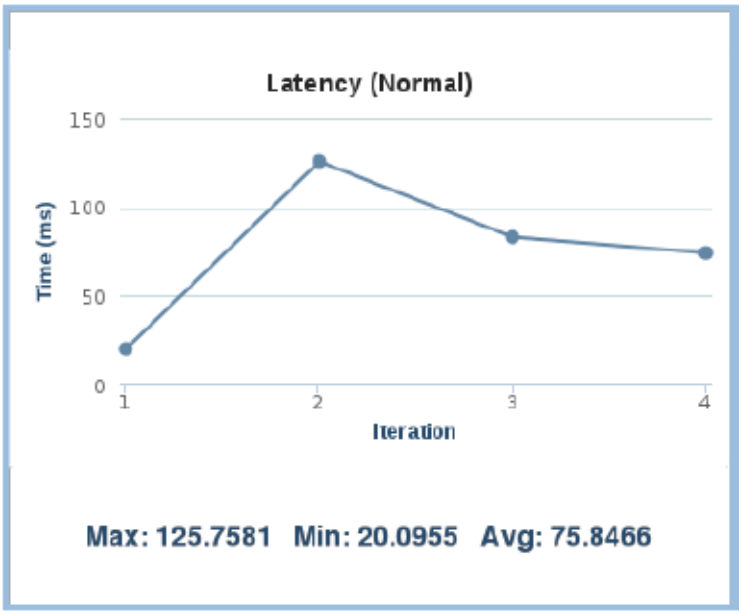
S.No	Iteration	Latency (ms)
1	1	19.8817
2	2	49.7340
3	3	269.1182
4	4	38.3441

Fig. 35: ONOS: 32 Switches (Normal)



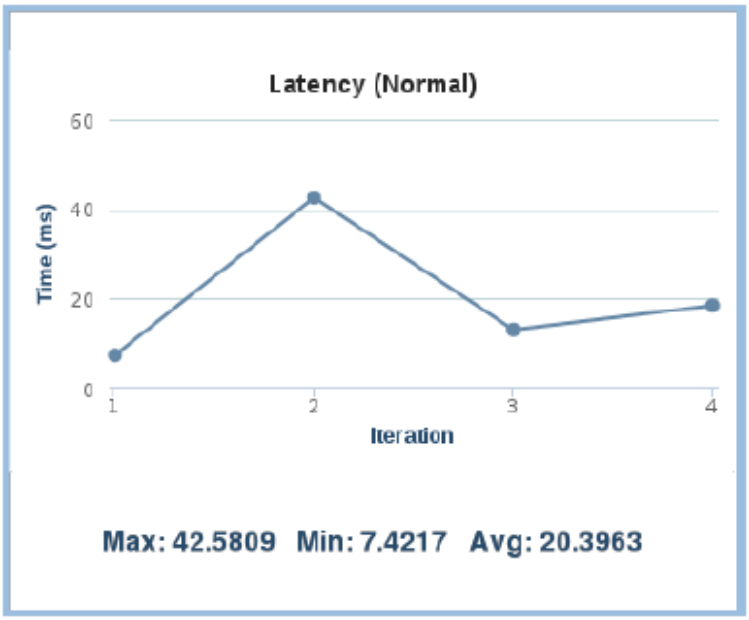
S.No	Iteration	Latency (ms)
1	1	101.8656
2	2	19.5589
3	3	11.4908
4	4	1.2230

Fig. 36: Floodlight: 32 Switches (Normal)



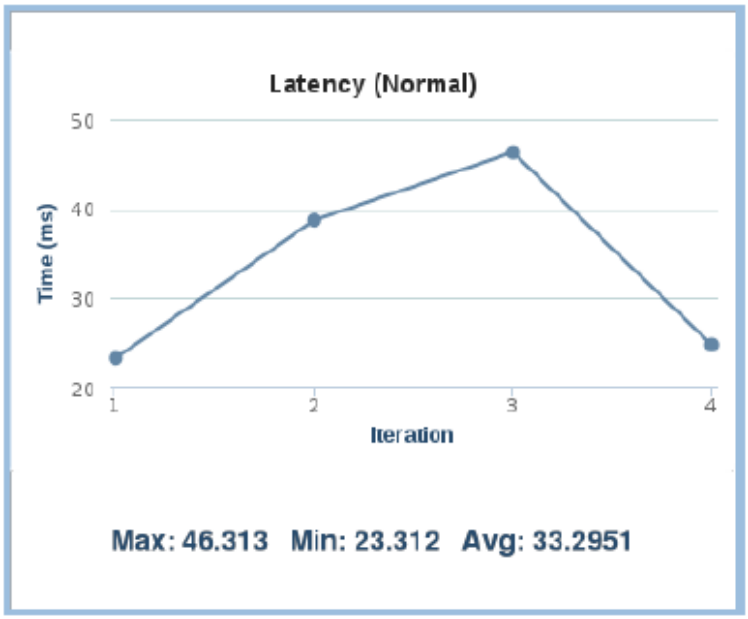
S.No	Iteration	Latency (ms)
1	1	20.0955
2	2	125.7581
3	3	83.2051
4	4	74.3277

Fig. 37: ONOS: 64 Switches (Normal)



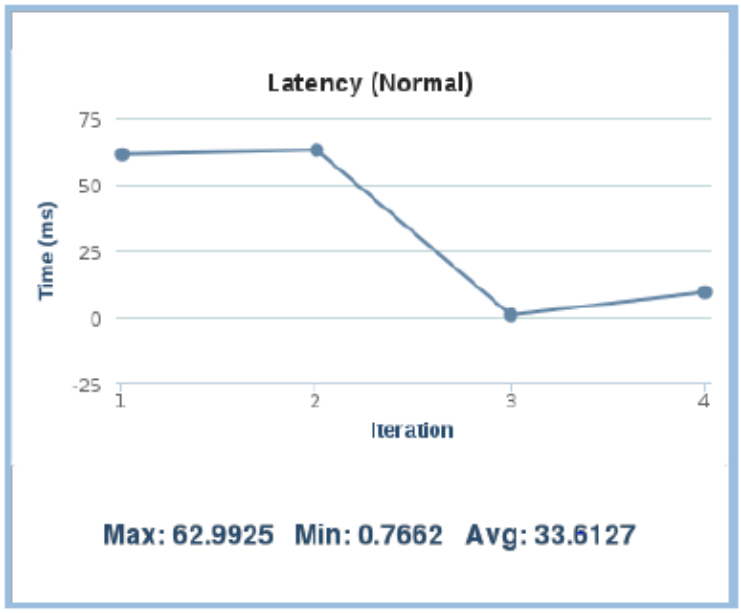
S.No	Iteration	Latency (ms)
1	1	7.4217
2	2	42.5809
3	3	13.1138
4	4	18.4689

Fig. 38: Floodlight: 64 Switches (Normal)



S.No	Iteration	Latency (ms)
1	1	23.3120
2	2	38.7331
3	3	46.3130
4	4	24.8222

Fig. 39: ONOS: 128 Switches (Normal)



S.No	Iteration	Latency (ms)
1	1	61.2866
2	2	62.9925
3	3	0.7662
4	4	9.4056

Fig. 40: Floodlight: 128 Switches (Normal)

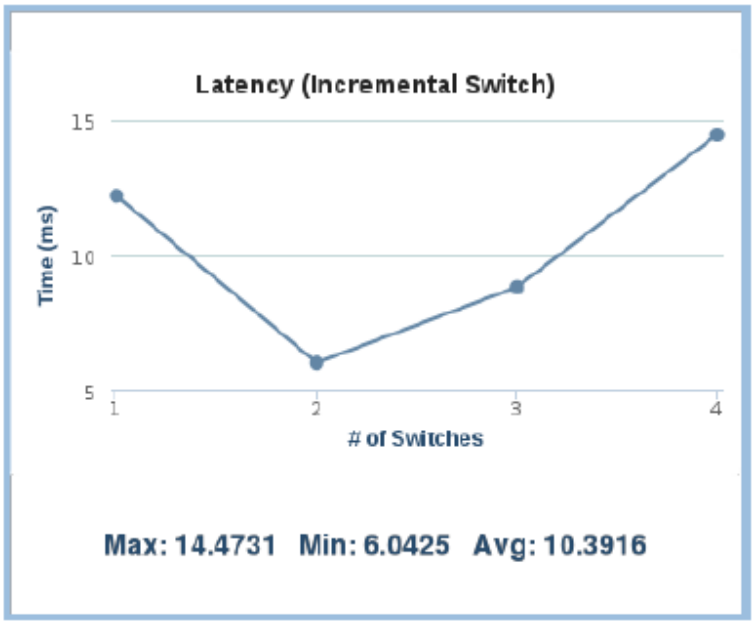
Performance Analysis of Latency Tests in Incremental Switch Mode

The Latency tests in Incremental Switch mode is executed in 4 iterations and in each iteration, 25% of the configured switches are added and in this way the load is increased by 25% for each iteration.

In this test, as we can see in the screenshots, Floodlight defeats ONOS with a big margin in all the 4 iterations for different loads of switches ranging from 4 to 128 switches.

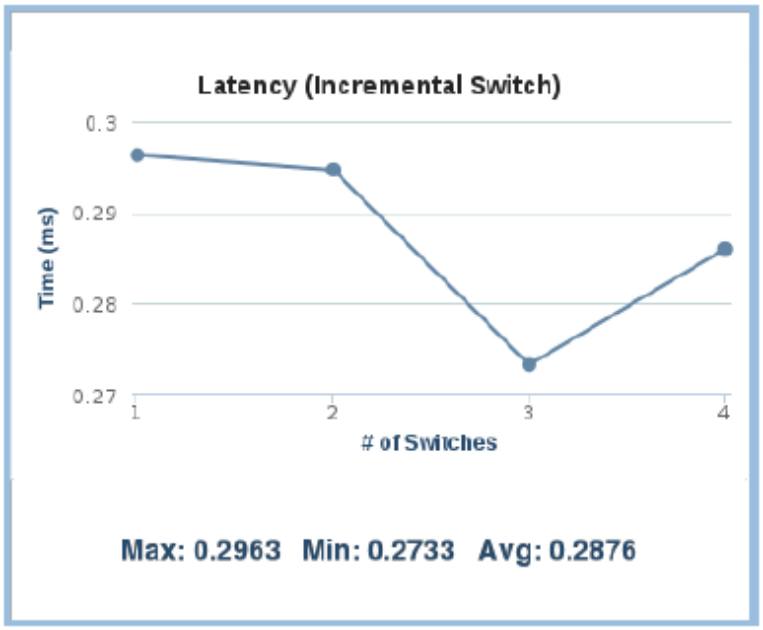
When the number of switches are increased, both the controllers start getting unstable. However, ONOS gets more unstable with higher number of switches compared to Floodlight.

Screenshots of the test results are attached in the following pages.



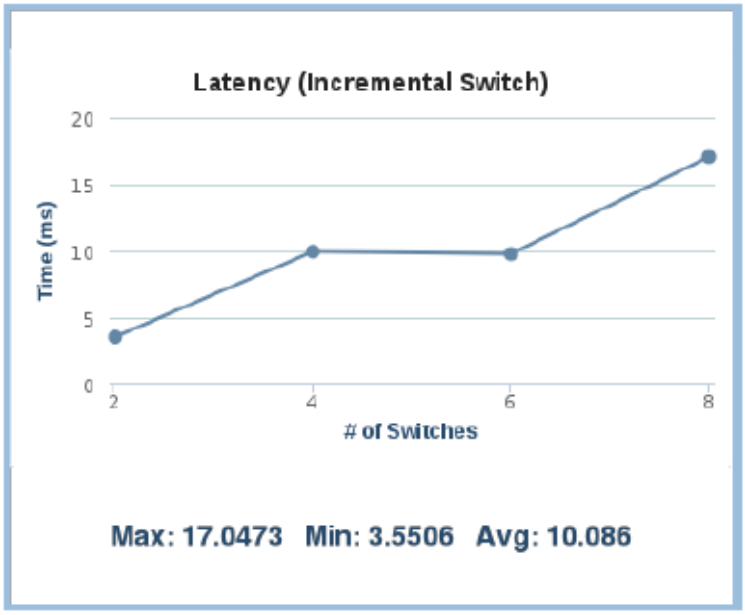
S.No	# of Switches	Latency (ms)
1	1	12.2163
2	2	6.0425
3	3	8.8343
4	4	14.4731

Fig. 41: ONOS: 4 Switches (Incremental Switch)



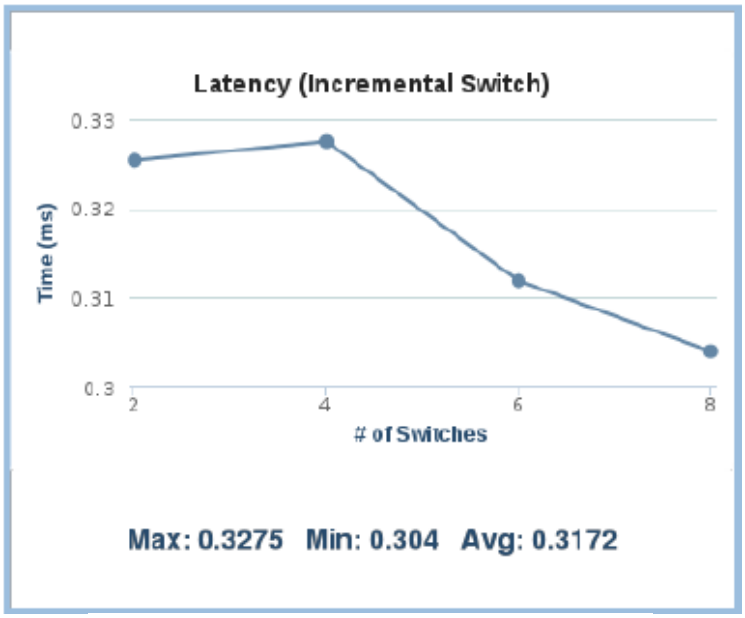
S.No	# of Switches	Latency (ms)
1	1	0.2963
2	2	0.2947
3	3	0.2733
4	4	0.2860

Fig. 42: Floodlight: 4 Switches (Incremental Switch)



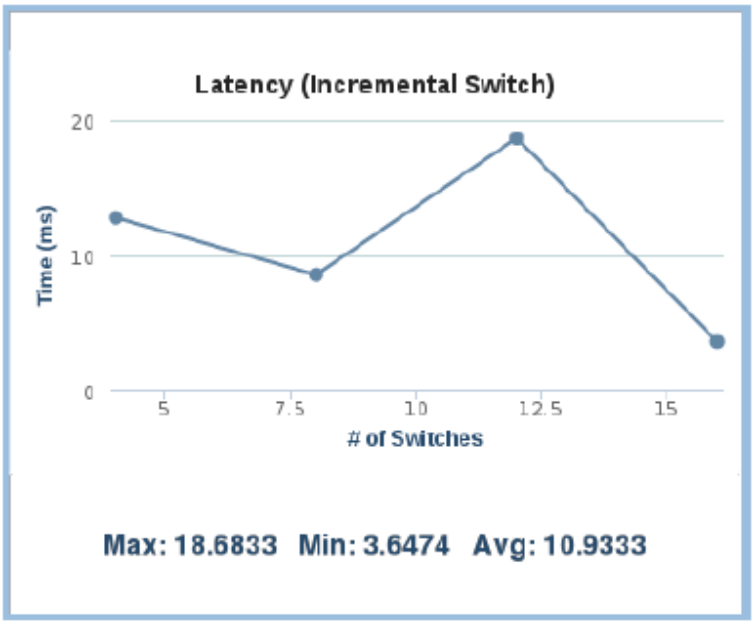
S.No	# of Switches	Latency (ms)
1	2	3.5506
2	4	9.9620
3	6	9.7841
4	8	17.0473

Fig. 43: ONOS: 8 Switches (Incremental Switch)



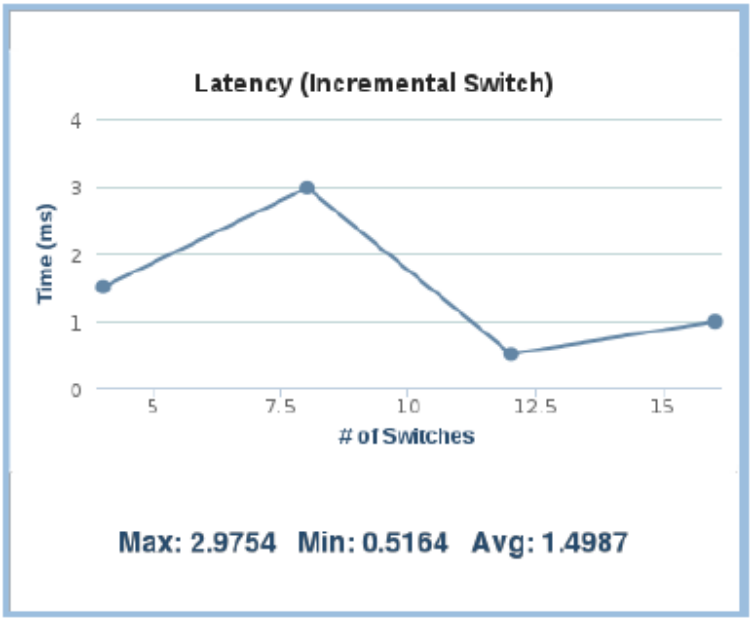
S.No	# of Switches	Latency (ms)
1	2	0.3254
2	4	0.3275
3	6	0.3119
4	8	0.3040

Fig. 44: Floodlight: 8 Switches (Incremental Switch)



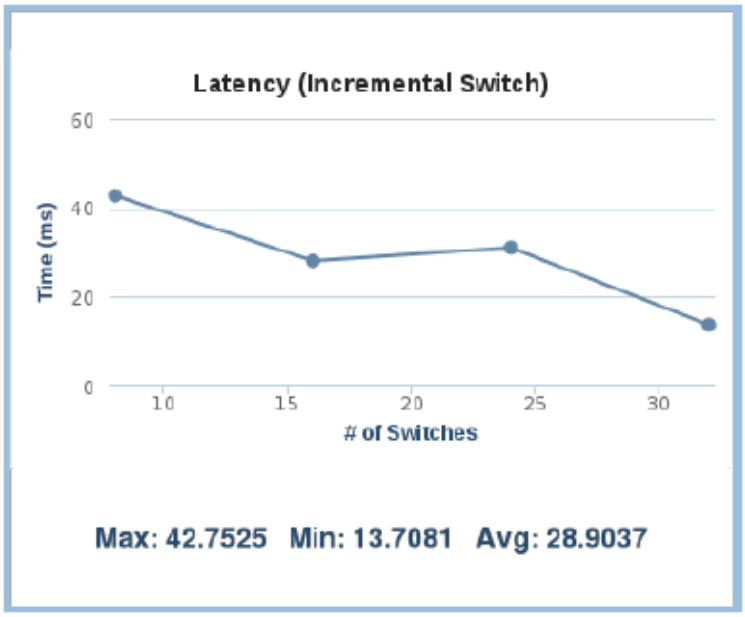
S.No	# of Switches	Latency (ms)
1	4	12.8040
2	8	8.5986
3	12	18.6833
4	16	3.6474

Fig. 45: ONOS: 16 Switches (Incremental Switch)



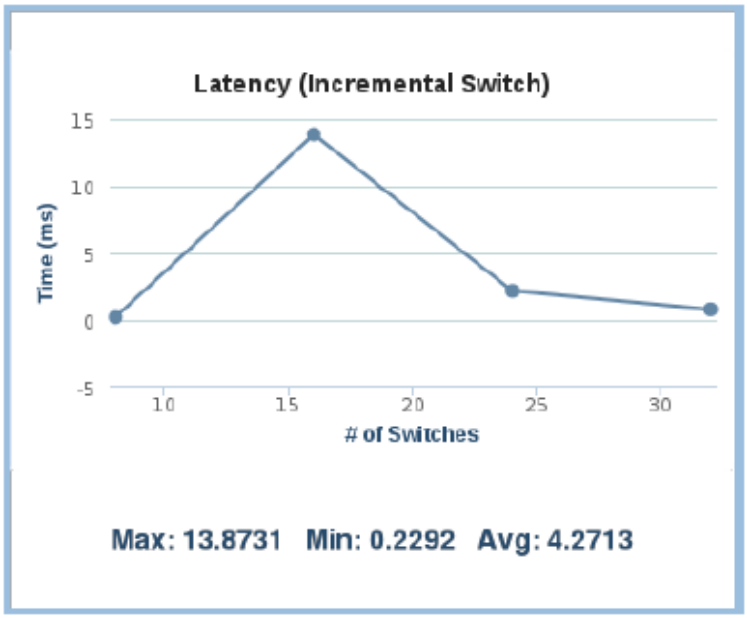
S.No	# of Switches	Latency (ms)
1	4	1.5075
2	8	2.9754
3	12	0.5164
4	16	0.9954

Fig. 46: Floodlight: 16 Switches (Incremental Switch)



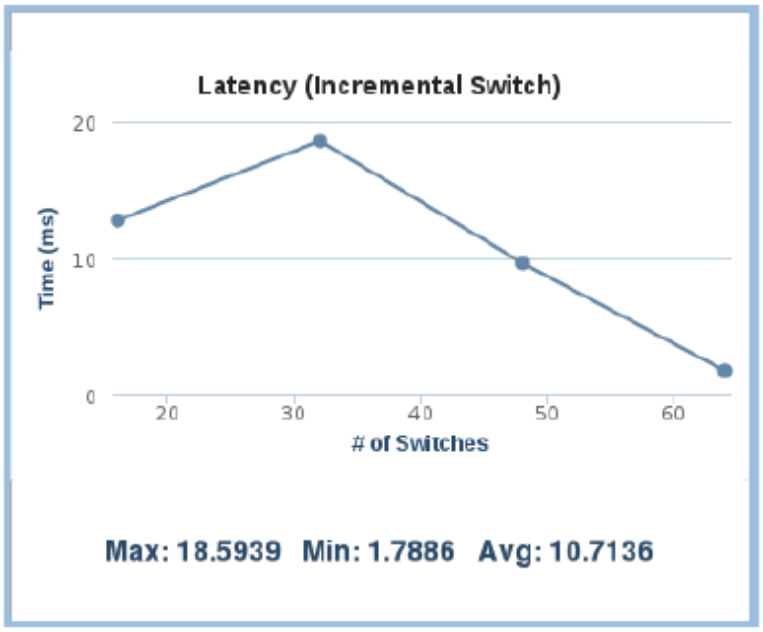
S.No	# of Switches	Latency (ms)
1	8	42.7525
2	16	28.0856
3	24	31.0685
4	32	13.7081

Fig. 47: ONOS: 32 Switches (Incremental Switch)



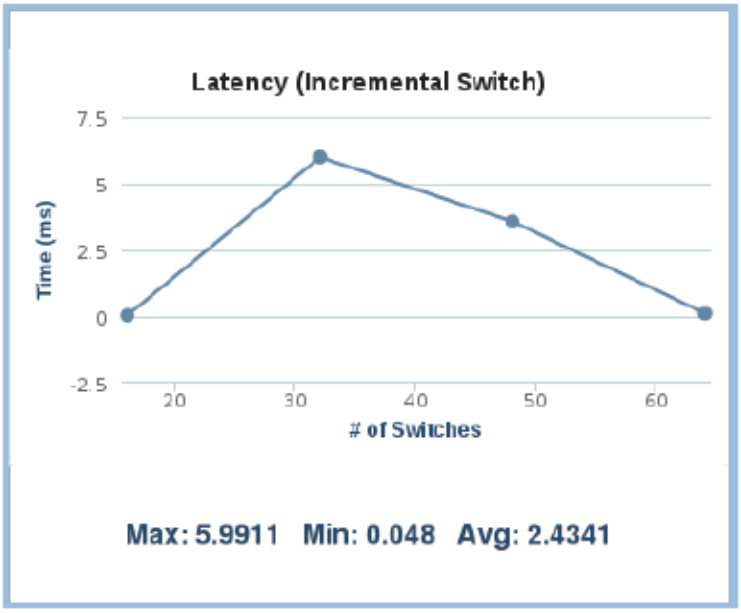
S.No	# of Switches	Latency (ms)
1	8	0.2292
2	16	13.8731
3	24	2.1844
4	32	0.7985

Fig. 48: Floodlight: 32 Switches (Incremental Switch)



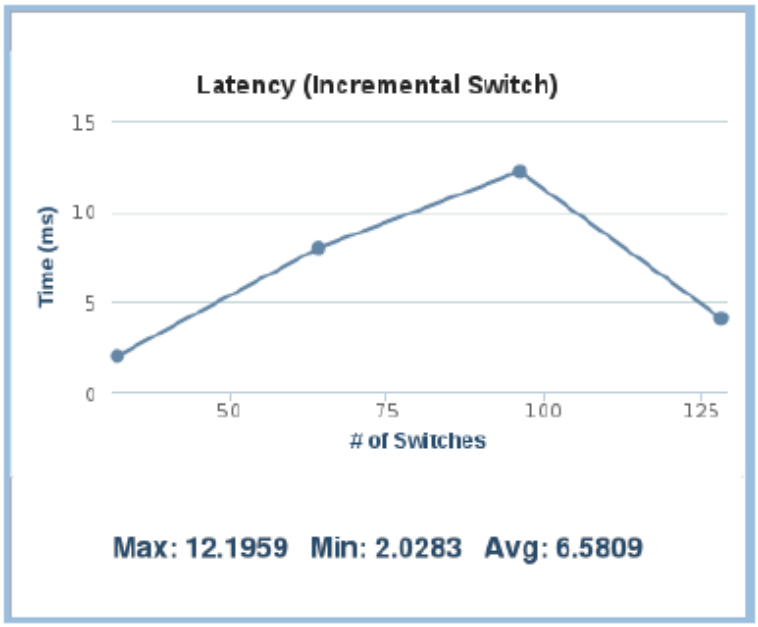
S.No	# of Switches	Latency (ms)
1	16	12.7945
2	32	18.5939
3	48	9.6773
4	64	1.7886

Fig. 49: ONOS: 64 Switches (Incremental Switch)



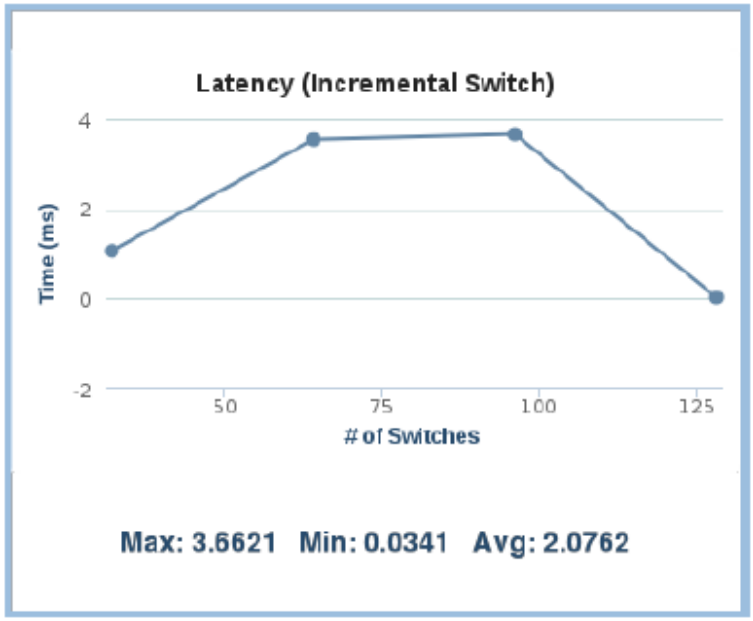
S.No	# of Switches	Latency (ms)
1	16	0.0480
2	32	5.9911
3	48	3.5806
4	64	0.1166

Fig. 50: Floodlight: 64 Switches (Incremental Switch)



S.No	# of Switches	Latency (ms)
1	32	2.0283
2	64	7.9806
3	96	12.1959
4	128	4.1189

Fig. 51: 128 Switches (Incremental Switch)



S.No	# of Switches	Latency (ms)
1	32	1.0682
2	64	3.5403
3	96	3.6621
4	128	0.0341

Fig. 52: Floodlight: 128 Switches (Incremental Switch)

Performance Analysis of Robustness Tests

A Robustness test tells the behavior of the controller. In this test, valid Packet-in messages are sent along with malformed Packet-in messages and robustness is measured based on the response of the controller (Flow-mod messages) to valid Packet-in messages.

This tool can measure Robustness for upto 64 switches.The test was performed for 4 iterations for different loads of switches upto 64 switches.

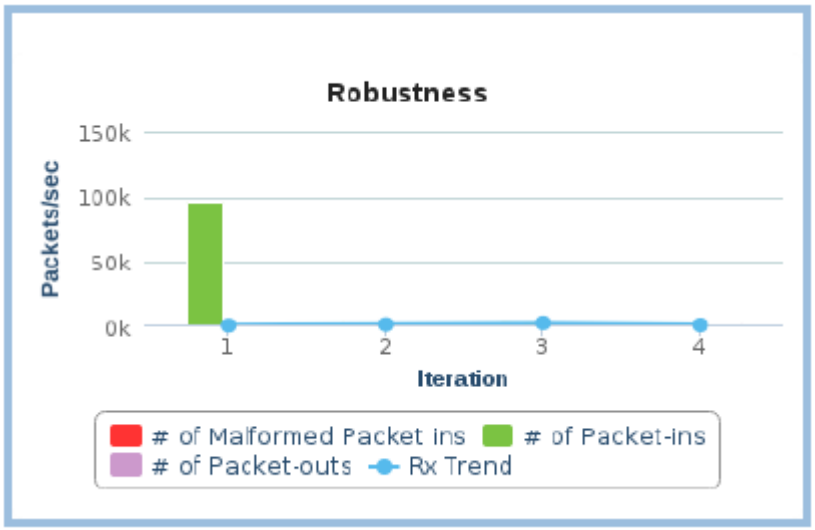
Based on the attached screenshots of test results in the following pages, It can be undoubtedly said that ONOS is far more robust compared to Floodlight.

All the tests show that in response to Packet-in messages along with Malformed Packet-in messages, ONOS responded with much higher number of Flow-mod messages as compared to Floodlight.

The point worth taking note here is that ONOS responded with Flow-mod messages in all the 4 iterations but Floodlight responded with Flow-mod messages only in 1 iteration out of the 4 in most the tests.

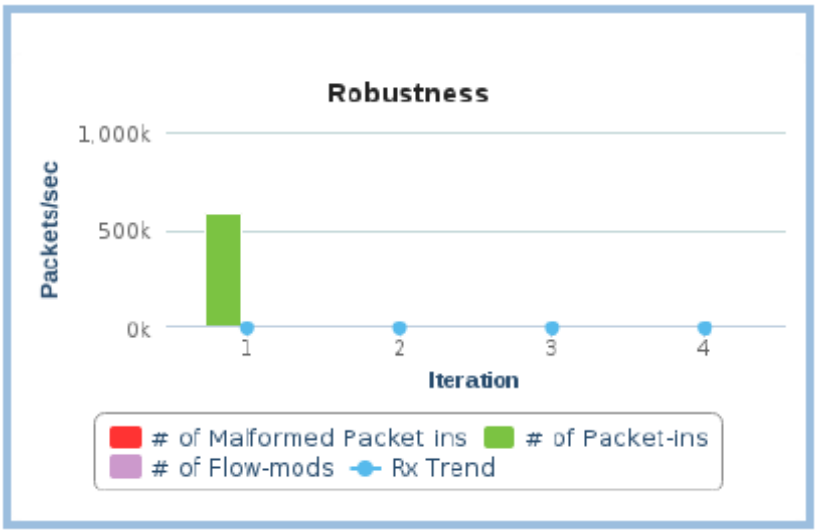
With 4 switches, Floodlight did not respond in either of the iterations. With 8 and 16 switches, it responded only in the 1st iteration and with 32 and 64 switches, it responded in 3rd and 2nd iteration respectively.

This clearly indicates that Floodlight is less robust as compared to ONOS. The gap between the two in the tests is very huge and Floodlight stands nowhere in front of ONOS in the robustness evaluation.



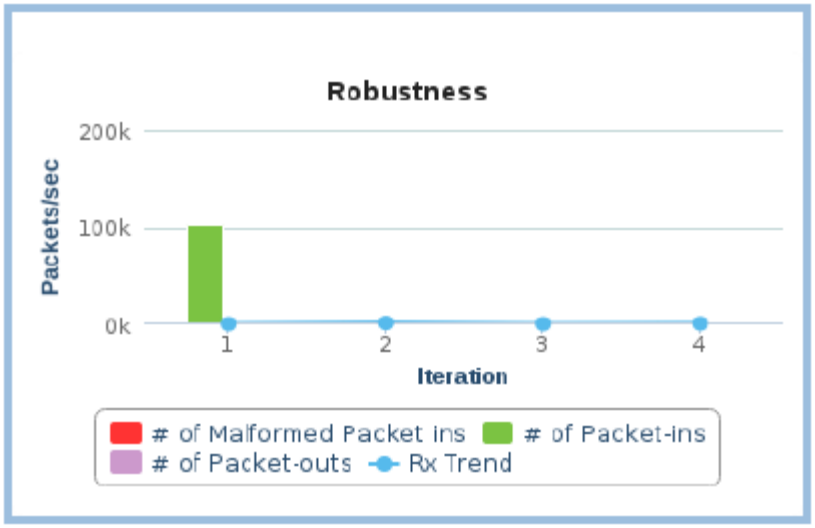
Iteration	% of Malformed Packets (Packet-in)	Total Tx Packets (Packet-in)	Tx Malformed Packets (Packet-in)	Total Rx Packets (Packet-out)
1	0	95,524	0	1,130
2	1	3,340	30	1,955
3	2	4,676	91	3,138
4	0	2,672	0	1,546

Fig. 53: ONOS: 4 Switches



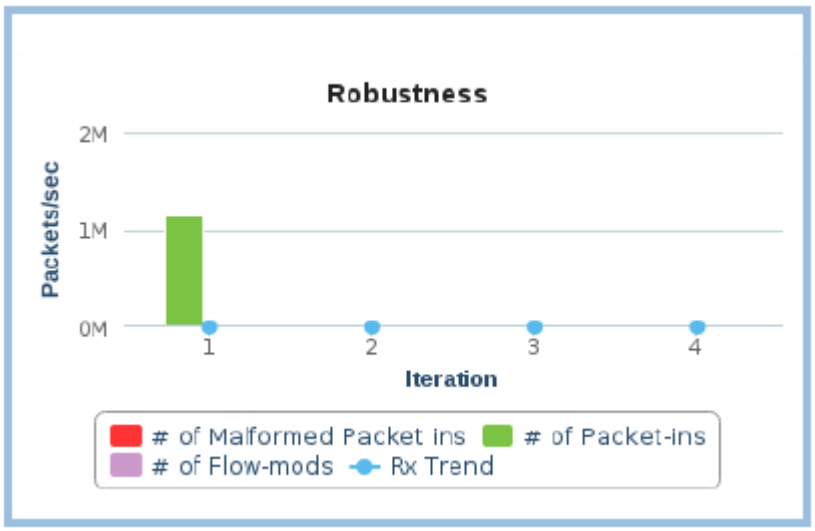
Iteration	% of Malformed Packets (Packet-in)	Total Tx Packets (Packet-in)	Tx Malformed Packets (Packet-in)	Total Rx Packets (Flow-Mod)
1	0	573,812	0	0
2	1	0	0	0
3	2	668	12	0
4	0	668	0	0

Fig. 54: Floodlight: 4 Switches



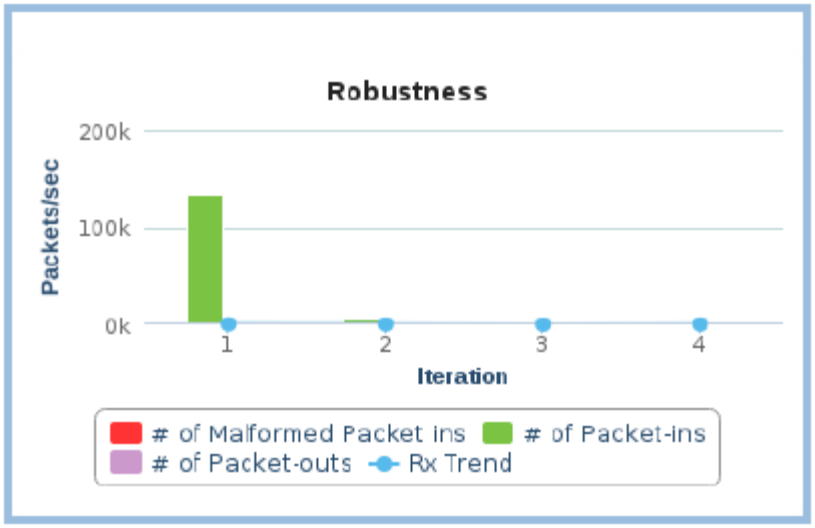
Iteration	% of Malformed Packets (Packet-in)	Total Tx Packets (Packet-in)	Tx Malformed Packets (Packet-in)	Total Rx Packets (Packet-out)
1	0	101,536	0	858
2	1	2,672	24	1,802
3	2	1,336	26	824
4	0	2,004	0	1,105

Fig. 55: ONOS: 8 Switches



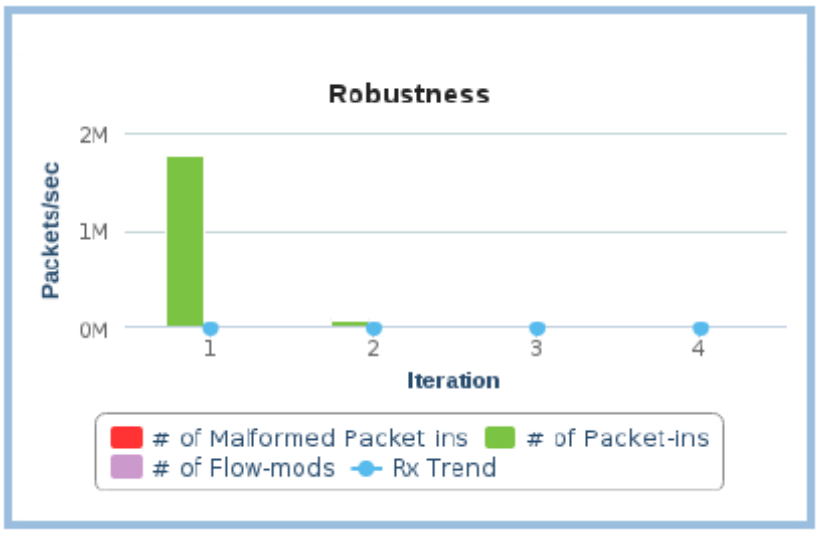
Iteration	% of Malformed Packets (Packet-in)	Total Tx Packets (Packet-in)	Tx Malformed Packets (Packet-in)	Total Rx Packets (Flow-Mod)
1	0	1,146,956	0	24
2	1	8,016	72	0
3	2	0	0	0
4	0	0	0	0

Fig. 56: Floodlight: 8 Switches



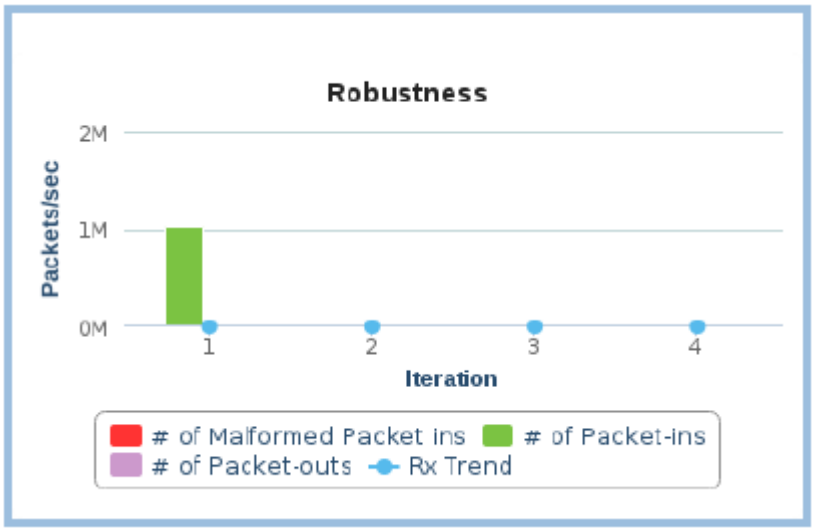
Iteration	% of Malformed Packets (Packet-in)	Total Tx Packets (Packet-in)	Tx Malformed Packets (Packet-in)	Total Rx Packets (Packet-out)
1	0	133,600	0	298
2	1	5,344	48	124
3	2	1,336	26	89
4	0	1,336	0	232

Fig. 57: ONOS: 16 Switches



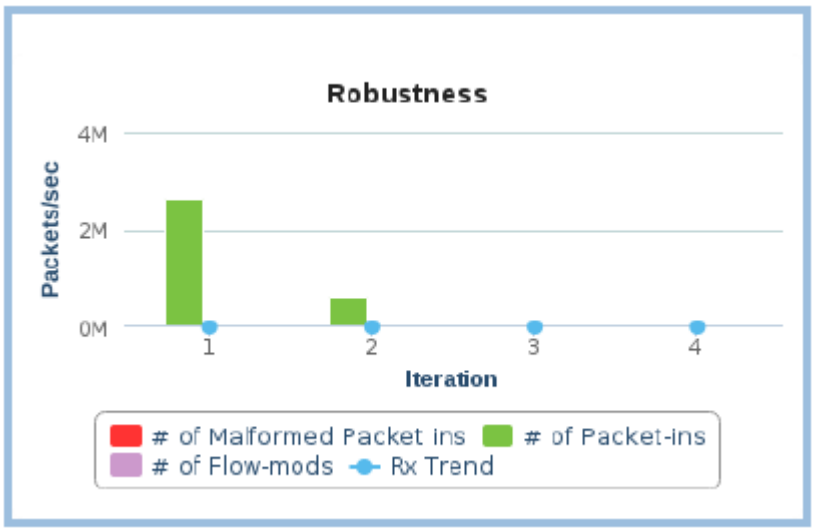
Iteration	% of Malformed Packets (Packet-in)	Total Tx Packets (Packet-in)	Tx Malformed Packets (Packet-in)	Total Rx Packets (Flow-Mod)
1	0	1,774,876	0	7
2	1	98,196	882	0
3	2	2,004	36	0
4	0	668	0	0

Fig. 58: Floodlight: 16 Switches



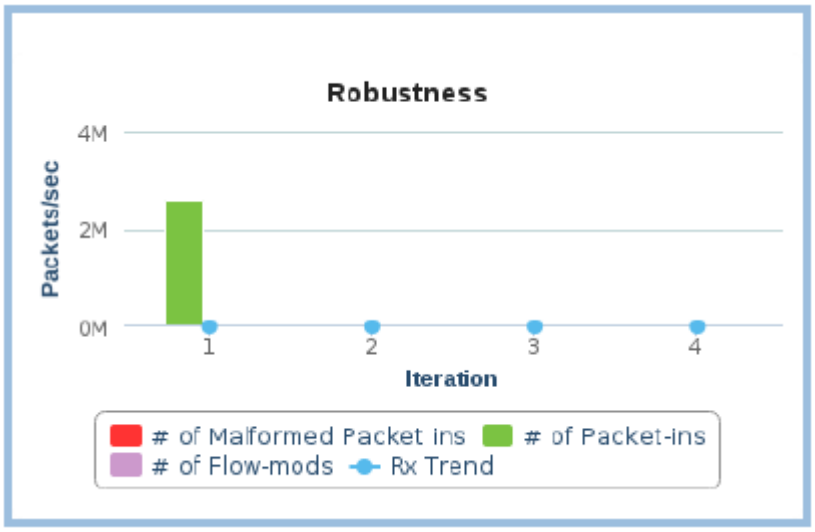
Iteration	% of Malformed Packets (Packet-in)	Total Tx Packets (Packet-in)	Tx Malformed Packets (Packet-in)	Total Rx Packets (Packet-out)
1	0	1,024,044	0	175
2	1	0	0	63
3	2	3,340	65	64
4	0	2,004	0	180

Fig. 59: ONOS: 32 Switches



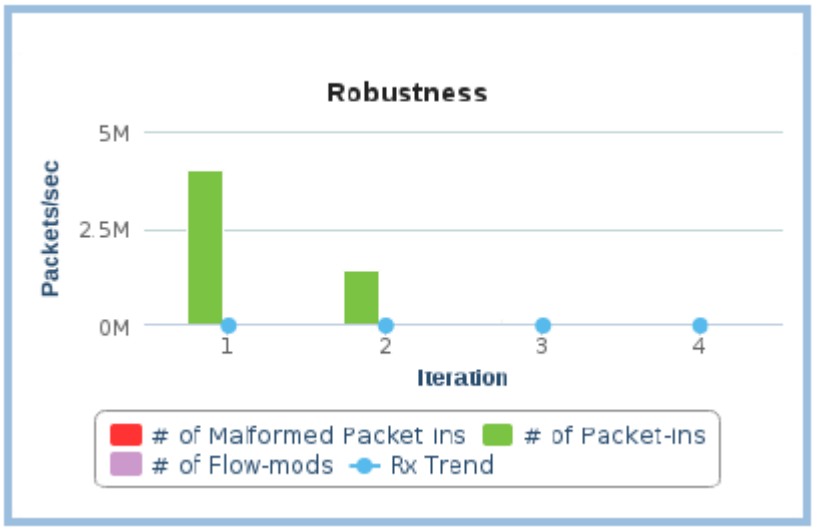
Iteration	% of Malformed Packets (Packet-in)	Total Tx Packets (Packet-in)	Tx Malformed Packets (Packet-in)	Total Rx Packets (Flow-Mod)
1	0	2,621,232	0	0
2	1	606,544	5,448	0
3	2	4,676	84	1
4	0	0	0	0

Fig. 60: Floodlight: 32 Switches



Iteration	% of Malformed Packets (Packet-in)	Total Tx Packets (Packet-in)	Tx Malformed Packets (Packet-in)	Total Rx Packets (Flow-Mod)
1	0	2,587,164	0	42
2	1	2,672	24	27
3	2	0	0	2
4	0	4,676	0	1

Fig. 61: ONOS: 64 Switches



Iteration	% of Malformed Packets (Packet-in)	Total Tx Packets (Packet-in)	Tx Malformed Packets (Packet-in)	Total Rx Packets (Flow-Mod)
1	0	3,981,948	0	0
2	1	1,417,496	12,732	60
3	2	97,528	1,752	0
4	0	12,692	0	0

Fig. 62: Floodlight: 64 Switches

Issues and Challenges

There were few major challenges and issues faced in the implementation of the project. The very first challenge was to select the versions of ONOS and Floodlight for comparing against each other. After doing detailed analysis, Version 1.5.1 (Falcon) of ONOS and Version 1.2 of Floodlight were chosen as they were released around same time and all the release notes were available for the two. ONOS was released in April 2016 while Floodlight was released in February 2016.

The other challenge was to build SDN controllers. Lots of steps are involved in building an SDN controller and one wrong step can make the build unsuccessful. ONOS was the most difficult to build out of the two controllers. A number of applications are required to install before building the controller like Java, Maven, Karaf, etc. and only the specific versions of these applications are compatible with the specific versions of ONOS. Installing other versions of these applications would result in errors.

Other major issue faced was with the performance benchmarking tool. Sometimes the tests were not able to complete and used to hang due to the unknown internal issues of the tool due to which the test had to be performed again.

Conclusion

Out of the three types of tests performed on the two controllers, ONOS emerged as the obvious winner in the Throughput and Robustness tests. It outperformed Floodlight heavily in the both sections. However, Floodlight left ONOS behind in terms of Latency. It was able to respond much more quickly to the configured switches as compared to ONOS.

These results conclude that for big IT organizations where there is a need to manage big network infrastructure, ONOS SDN controller would serve as an ideal and a better option as compared to Floodlight controller.

Throughput and Robustness are more important parameters as compared to latency. Latency is not going to make a very big difference as it is a matter of milliseconds which would have a negligible effect on the overall functioning of the network infrastructure.

As the tests show, Both ONOS and Floodlight were able to respond to the Packet-in messages of the switches in milliseconds with Floodlight being little faster in responding as compared to ONOS. However, taking the other two parameters into consideration i.e. Throughput and Robustness, the gap between the two is huge and Floodlight is far behind that ONOS which concludes ONOS could be a more reliable SDN controller than Floodlight.

References

[1]https://www.researchgate.net/publication/262240269_Performance_Analysis_of_Software_Defined_Networking_SDN

[2]http://thenewstack.io/sdn-series-part-eight-comparison-of-open-source-sdn-controllers/

[3] https://www.youtube.com/watch?v=Q0CSA9xE7rU

[4]http://sdn.veryxtech.com/download/PktBlaster-SDN_Controller_Test-Performance_Benchmarking_UserGuide_1.4.pdf

[5]https://en.wikipedia.org/wiki/ONOS

[6]http://www.prnewswire.com/news-releases/onlab-delivers-software-for-new-open-source-sdn-network-operating-system--onos-300004797.html

[7]https://wiki.onosproject.org/display/ONOS/System+Components

[8]http://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343549/Architecture

[9]https://www.sdxcentral.com/sdn/definitions/sdn-controllers/open-source-sdn-controllers/what-is-floodlight-controller/

[10]http://www.projectfloodlight.org/floodlight/

[11] https://pdfs.semanticscholar.org/47f8/1a0fe08310cc732fbd6ad16ab9da323f395c.pdf

[12]Benchmarking Methodology for SDN Controller Performance
https://tools.ietf.org/html/draft-bhuvan-bmwg-sdn-controller-benchmark-meth-00

[13]Software-Defined Networking: A Comprehensive Survey
https://arxiv.org/pdf/1406.0440.pdf

Table of Figures