Performance Comparison of Ryu and Floodlight Controllers in Different SDN Topologies

Ravindra Kumar Chouhan
Department of Computer Applications
NIT Raipur
Raipur, India
Email: rchauhan@nitrr.ac.in

Mithilesh Atulkar
Department of Computer Applications
NIT Raipur
Raipur, India
Email: matulkar.mca@nitrr.ac.in

Naresh Kumar Nagwani
Department of Computer Science
NIT Raipur
Raipur, India
Email: nknagwani.cs@nitrr.ac.in

Abstract— **Ryu and Floodlight controllers are being used in academia and industries for different research and development purposes. The objective of this paper is to compare the performance of these controllers under different topologies namely single, linear, tree, torus and custom. Experiments are performed under a much known emulator, mininet. Four parameters were valuated under this experimental setups namely throughput, jitter, latency and packet loss. Valuation of results clearly shows that Ryu is giving better throughput as compared to Floodlight in all the topologies. In case of latency and jitter Ryu is performing better in all the topologies except Torus. Similar is the case with packet loss.**

Keywords— **SDN, Openflow, Ryu, Floodlight, Mininet**

I. INTRODUCTION

As the SDN is being used in many research and development purposes in academia and industries, its performance must be very good to control the overall network of an organization. The performance of SDN basically depends upon the performance of the controller used in that SDN as controller is the main component of as SDN and the overall responsibility of running a network efficiently is its main objective. In such scenario it is always advisable to select a good controller as per the need. How to select the best controller for our need will depend upon the performance evaluation of the controllers being considered for our work.

Currently around more than 30 controllers have been proposed by both industries and academics. These controllers have been developed in different languages, different run time technologies and with different features. Selecting one controller among these available controllers requires performance evaluation of the controllers.

Many works have been done in this connection and results have been published but they are lacking some more parameters to be considered for the performance evaluation. In this paper two controllers have been taken for performance comparison namely Ryu and Flood Light. Reason behind selecting these two controllers is that Ryu performed best when compared with other python based controllers [1] and Floodlight, a Java based controller, performs better when compared with ROX, a python based controller[2]. Moreover, their application domain is same i.e. campus [3] and they are open source controller [4]. So there is a need to check whether Ryu (Best controller among python based controllers) is better or worse than Floodlight in performance.

In this paper, four broadly used parameters to measure the performance of a network namely throughput, jitter, latency and packet-loss have been compared in different topologies namely single, linear, tree, torus and custom. To check the throughput, jitter and packet-loss of the network Iperf has been used and to check latency, Round Trip Time (RTT) has been used.

The paper is further organized as follows: Section 1.2 discusses the previous works done for comparing the performance of the SDN controllers, Section 1.3 compares the different features of Ryu and Floodlight (controllers under the consideration), Section 1.4 shows our contribution, Section 1.5 shows how the experimental environment has been set up and experiments performed. Section 1.6 and 1.7 show result analysis and conclusion & future work respectively.

II. RELATED WORKS

The performance study of four controllers namely POX, Ryu, ONOS and OpenDaylight has been done by Alexandru L. Stancu et al. [5]. In their setup they compared the TCP bandwidth of all the mentioned controllers on switch and hub mode. They found that bandwidth in switch mode is quite big in Gbps as compared to that in hub mode that is in Mbps and Kbps. They found ONOS the best controller among mentioned controller. While working on RTT they found that ONOS has the lowest convergence time. They concluded with not mentioning any controller good or bad but stated it depends on the needs and situations of the networks. They also stated that this type of comparison can be improved by considering even more implementation such as Floodlight and Trema.

Work done in [6] is to compare the performance of five SDN controllers namely Ryu, POX, Nox, Floodlight and Beacon. They have observed latency, throughput and the performance of controllers on the basis of "responses per second" using cbench. They found Beacon's performance is best under all these parameters and said Ryu as competitive due to its active development community. They also changed the number of nodes and threads many times and found Beacon's performance best. They have fixed the topologies and only changed the switch numbers. There is a need to change the topology also.

The performance study of three python based controllers namely POX, Ryu and Pyretic has been done by Karamjeet Kaur et al. [1]. They used Iperf for performance measurement and RTT and Web Server Latency for latency comparison. They found in all the cases Ryu is the best. They did not check other parameters and also did not check for other topologies.

The performance of mininet's reference controller using four parameters namely bandwidth utilization, packet transmission rate, Round Trip Time, and throughput on three topologies namely single, linear and tree has been done by Idris Zoher Bholebawa et al. [7]. They found that reference controller performs best among said three. Reason behind this is that now all the entries of openflow are in a single switch. They have not evaluated different topologies on different controllers.

In [2] study has been done on two controllers namely POX (a python based controller) and Floodlight (a Java based controller). They evaluated the throughput and RTT on four topologies namely single, linear, tree and custom. They found that throughput and RTT of Floodlight is better than that of POX.

The performance study of four controllers; Ryu, Floodlight, ONOS and OpenDaylight based on two parameters; throughput and latency has been done by Lusani Mamushiane et al. [3]. To perform the test they have used cbench. In their practical setup they performed the test by changing number of switches in one scenario and number of MACs in another scenario. They concluded that ONOS is performing best in case of throughput and Ryu is performing best in case of least latency. But they did not valuate by changing the topologies.

In [8] performance analysis of POX and Ryu has been done on different topologies namely Single, Linear, Tree, Dumbbell, DCN and SDN-SAT on two parameters namely throughput and latency. They concluded that Ryu is better than POX.

III. MOTIVATION

It is found through the above papers that to evaluate the performance comparison authors are either fixing topology and changing switch count or if they are changing topologies they are not checking them on different controllers. Also there is a need to compare Ryu the best controller among python based controller with Floodlight a Java based controller as both can be used for a campus size network and both are open source controllers. Also there is need to compare some more network parameters on these two controllers to get the better conclusion.

IV. COMPARISON OF RYU AND FLOODLIGHT

A. Ryu

Ryu [10] pronounced as 'ree-yooh' is an opensource controller for SDN. It supports many protocols needed for managing networking devices in SDN like Openflow, Netconf, OF-config, etc. it supports all versions of Openflow 1.0, 1.2, 1.3, 1.4. 1.5 and Nicira Extensions. It provides very well defined APIs which make it easier for developers to create network management and control application. Its code is freely available under the Apache 2.0 license.

B. Floodlight

The Floodlight Open SDN Controller [11] 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.

TABLE I. FEATURE COMPARISON OF RYU AND FLOODLIGHT

Feature	Ryu	Floodlight
Southbound Interfaces	Openflow 1.0, 1.2, 1.3, 1.4, 1.5, Nicira Extensions	Openflow 1.0, 1.1, 1.2, 1.3, 1.4, 1.5
REST API	Yes (For SB Only)	Yes
GUI	Yes(Initial Phase)	Web/ Java -based
Modularity	Medium	Medium
Orchestrator Support	Yes	Yes
OS Support	Most supported on Linux	Linux, Windows and MAC
Partner	Nippo Telegraph and Telephone Corporation(NTT)	Big Switch Network
Documentation	Medium	Good
Programming Language	Python	Java
Multi threading Support	Yes	Yes
TLS Support	Yes	Yes
Virtualization	Mininnet and OVS	Mininet and OVS
Application Domain	Campus	Campus
Distributed/Centralized	Centralized	Centralized

V. IMPLEMENTATION

A. Emulation Environment

Mininet [9] and controller both were run on a machine having 8 GB RAM, Ubuntu 16.04.5 LTS 64 bit Operating system and AMD PRO A8-8650B R7, 10 Compute Cores 4C+6G @ **Turbo Speed:** *3.9 Ghz* Processor.

B. Topologies

Topology represents the physical connectivity/arrangements of the network devices/nodes. In the emulation environment five topologies were implemented single, linear, tree, torus and custom. The number of hosts and switches connected in different topologies are given in Table II.

TABLE II. NUMBER OF HOSTS AND SWITCHES CONNECTED IN DIFFERENT TOPOLOGIES

Topology	No of hosts	No of Switches
Single	100	1
Linear	100	100
Tree	64	9
Torus	9	9
Custom	32	5

a) Single: In single topology only one switch is used and all the hosts are connected to that switch. The emulated single network is shown Fig. 1.

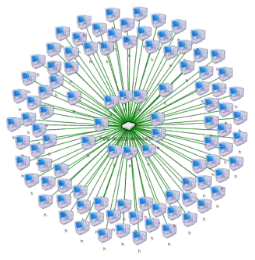

Fig. 1. Single topology generated by Floodlight Controller

*b) Linear**:*** Linear topology is like bus topology where each openflow enabled switch are connected in linear manner. The emulated Linear network is shown Fig. 2.

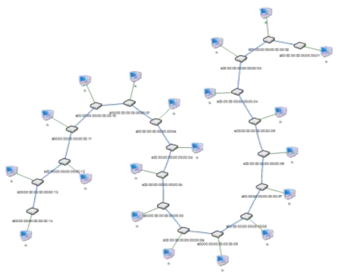

Fig. 2. Linear topology generated by Floodlight Controller

c) Tree: There are two terms associated with tree topology depth and fanout. Depth represents the number of levels of switch and fanouts are the number of output ports where switch/hosts will connect. The emulated Tree network is shown in Fig. 3.

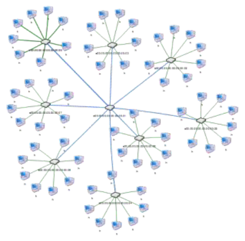

Fig. 3. Tree topology generated by Floodlight Controller

d) ***Torus:*** In torus topology switches are connected to its adjacent neighbour switch and the farthest switch. Farthest switches are connected so that connectivity could be provided in cyclic way. The emulated Torus network is shown in Fig. 4.

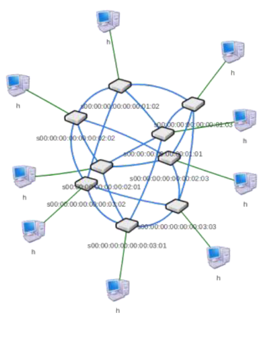

Fig. 4. Torus topology generated by Floodlight Controller

e) ***Custom:*** In custom topology there is one top level switch and four switches are connected with this top level switch. Now 8 hosts are connected to each lower level switch thus total 64 switches are connected. The emulated custom network is shown in Fig. 5.

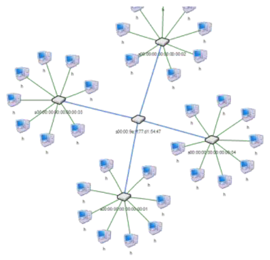

Fig. 5. Custom topology generated by Floodlight Controller

VI. RESULT ANALYSIS

The above mentioned topologies namely single; linear*,* tree, torus and custom were created in the mininet and connected with Ryu and Floodlight controllers one by one. When one topology is connected with a particular controller then Iperf [12] test was performed to measure the throughput, jitter and packet-loss by taking the most distant hosts. Between two most distant hosts last one was treated as Iperf server and first one was treated as Iperf client. To measure throughput, Iperf server was run in TCP mode. 20 TCP packets were sent from client to server and the throughput was observed. To calculate Jitter and data loss, server was run in UDP mode and now 20 UDP packets from the clients were sent and Jitter and data loss was calculated. To calculate the latency 15 packets from the client were sent to server and the average RTT was taken as the latency of the network.

Experiments are performed and results are presented in Fig. 6, Fig. 7, Fig. 8 and Fig. 9.

From the graph shown in Fig. 6, it is clear that throughput of Ryu is better than that of Floodlight in all the topologies under consideration. Throughput of Ryu is 9.83%, 85.72%, 4.19%, 8.47% and 7.28% higher than that of Floodlight controller in custom, linear, single, torus and tree topologies respectively.

From the graph shown in Fig. 7, it is clear that Latency in Ryu controller is less than that of Floodlight controller in all the topologies except in Torus. Ryu controller is providing 74.65%, 92.92%, 74.17%, 75.64% less latency than Floodlight in Custom, linear, single, and tree topologies. Floodlight is providing less latency in torus topology only and it is only 30% high.

From the graph shown in Fig. 8, it is clear that Jitter in Ryu controller is less than that of Floodlight controller in all the topologies except in Torus. Ryu controller is providing 52.84%, 74.06%, 30.74%, 17.97% less jitter than Floodlight in Custom, linear, single, and tree topologies. Floodlight is providing less jitter in torus topology.

Reason behind the high latency and jitter of Ryu in Torus topology might be that switches were not connected directly in torus topology but they were connected through bridges.

From the graph shown in Fig. 9, it is clear that packet-loss percent in Floodlight controller is higher than Ryu except in linear topology.

Summarily, it can easily be stated that performance of Ryu is better than that of Floodlight in most of the cases. Ryu provides better throughput in all the topologies. Its latency and jitter are more than Floodlight in only torus topology but in remaining topologies namely single, linear, tree and custom, it provides less latency and jitter than Floodlight. In case of packet-loss also, performance of Ryu is better than Floodlight in all topologies except in linear topology.

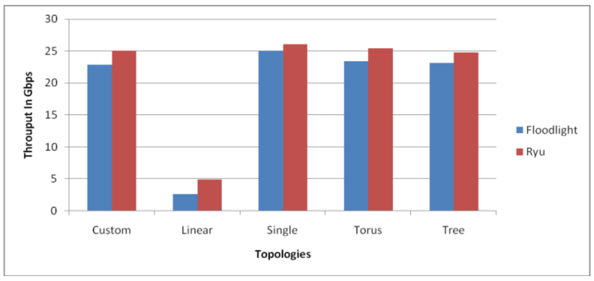

Fig. 6. Throughput Comparison of Ryu and Floodlight

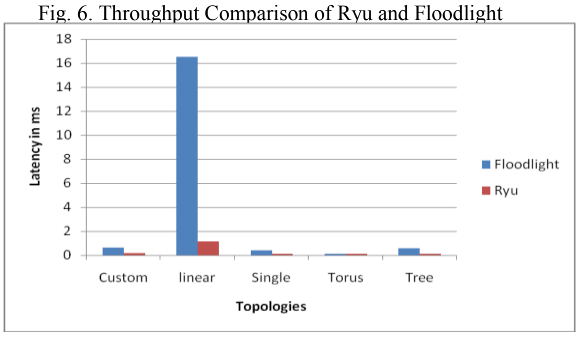

Fig.7. Latency Comparison of Ryu and Floodlight

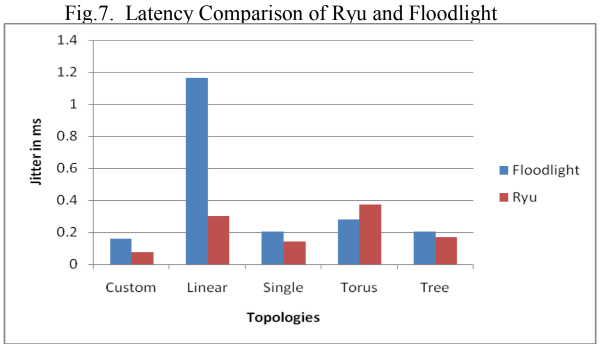

Fig. 8. Jitter Comparison of Ryu and Floodlight

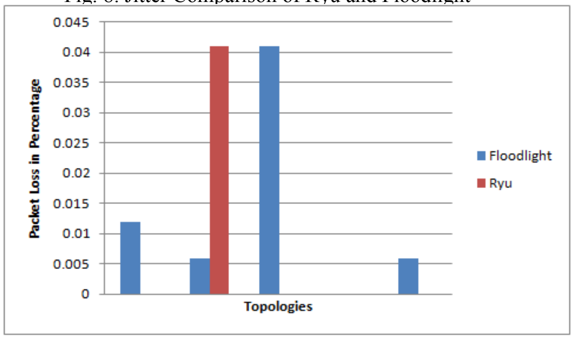

Fig. 9. Packet-loss comparison in Ryu and Floodlight

VII. CONCLUSION AND FUTURE WORK

By evaluating the throughput, latency, packet-loss and jitter of Ryu and Floodlight it can easily be stated that performance of Ryu is better than that of Floodlight in most of the cases. Ryu provides better throughput in all the topologies. Its latency and jitter are more than Floodlight in only torus topology but in remaining topologies namely single, linear, tree and custom, it provides less latency and jitter than Floodlight. In case of packet-loss also, performance of Ryu is better that Floodlight as its packet-loss rate is lower than that of Floodlight in all topologies except in linear topology. It is also observed that code updation for research purpose in Ryu is easily possible whereas this is quite difficult in Floodlight controller. In future the same work can be performed on the controllers by sending malpackets in the network to check their security and robustness. Also the same work can be performed on complex networks especially on Data Center Networks.

REFERENCES

[1] Kaur, K., Kaur, S. and Gupta, V. ; Performance analysis of python based openflow controllers. 3rd International Conference on Electrical, Electronics, Engineering Trends, Communication, Optimization and Sciences (EEECOS 2016)., 2016, Tadepalligudem, India.

[2] Bholebawa, I. and Dalal, U. Performance Analysis of SDN/OpenFlow Controllers: POX Versus Floodlight. Wireless Personal Communications, 98(2), pp.1679-1699, Jan. 2017.

[3] Mamushiane, L., Lysko, A. and Dlamini, S. A comparative evaluation of the performance of popular SDN controllers. 2018 Wireless Days (WD). 2018 Dubai, United Arab Emirates.

[4] Khondoker, R., Zaalouk, A., Marx, R. and Bayarou, K. (2014). Feature-based comparison and selection of Software Defined Networking (SDN) controllers. 2014 World Congress on Computer Applications and Information Systems (WCCAIS). Jan. 2014, Hammamet, Tunisia.

[5] Stancu, A., Halunga, S., Vulpe, A., Suciu, G., Fratu, O. and Popovici, E. (2015). A comparison between several Software Defined Networking controllers. 2015 12th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS), Oct. 2015, Nis, Serbia.

[6] Zhao, Y., Iannone, L. and Riguidel, M. (2015). On the performance of SDN controllers: A reality check. 2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)., Jan. 2016, San Francisco, CA, USA.

[7] Bholebawa, I. and Dalal, U. (2016). Design and Performance Analysis of OpenFlow-Enabled Network Topologies Using Mininet. International Journal of Computer and Communication Engineering, 5(6), pp.419-429, Nov 2016.

[8] Ali, J., Lee, S. and Roh, B. (2018). Performance Analysis of POX and Ryu with Different SDN Topologies. Proceedings of the 2018 International Conference on Information Science and System - ICISS '18. pp 244-249, Apr, 2018, Jeju, Republic of Korea.

[9] de Oliveira, R., Schweitzer, C., Shinoda, A. and Ligia Rodrigues Prete (2014). Using Mininet for emulation and prototyping Software-Defined Networks. 2014 IEEE Colombian Conference on Communications and Computing (COLCOM). Bogota, Colombia, Jun 2014.

[10] Ryu Controller, available at - https://osrg.github.io/Ryu/ (Last accessed on 10-11-2018).

[11] Floodlight Controller, available at - http://www.projectfloodlight.org/floodlight/ (Last accessed on 10-11-2018).

[12] Iperf , available at - https://iperf.fr/ (Last accessed on 10-11-2018).