International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org, editorijettcs@gmail.com
Volume XX, Issue XX, May-June 2022 ISSN 2278-6856

Comparative Assessment of SDN Openflow Controllers under Mininet Emulation Environment

Arvind T[1] and Dr.K.Radhika[2]

[1]Research Scholar, Department of CSE, UCE, OU, Hyderabad

[2]*Professor, Department of Information Technology, CBIT, Hyderabad*

Abstract: ***Software Defined Network (SDN) is an emerging technology in the field of cloud computing and networking, which simplifies the tasks of the network administrators, by providing centralized management of networking devices,by separation of data and control planes. Various cloud computing environments, enterprise data centres and service providers are utilizing this essential feature. In this work we have employed mininet to illustrate the aptness of SDN for different scalability. We examined the performance of SDN controllers such as Ryu, POX and Floodlight controllers utilizing mininet and qperf, a network performance measurement tool. During this study we have utilized different network topologies with different number of switches and hosts to assess the performance of the controllers. The performance parameters used were network latency and bandwidth. Experimental results demonstrated that Ryu provides better bandwidth and minimum latency in all the topologies, except single topology. POX controller outperformed Ryu and Floodlight controllers in single topology***

Keywords: SDN, Mininet, Ryu, POX, Floodlight, qperf.

1. INTRODUCTION

Software-defined networking is a framework in which the network devices are managed by means of a centralized controller, which has a global perspective of the network and implements policy level decisions to the relevant network elements, by using high level languages and API's. It therefore optimizes the efficiency and adaptability of the network.

SDN Architecture:

The core objective of the SDN architecture is to detach the control plane from the underlying network devices, namely routers and switches that forward the network packets. Thus decoupling, makes the network devices behave as dumb devices, which act consistent with the flow rules directed by the centralized controller [1-3]. Routing decisions are made on flow rules rather than source-destination based. A software defined network consists of three layers or planes such as Application plane, Control plane and Data plane, which are connected using required interfaces [32].

1. Application layer/plane: This is the topmost layer in SDN architecture. It deals with the end user business

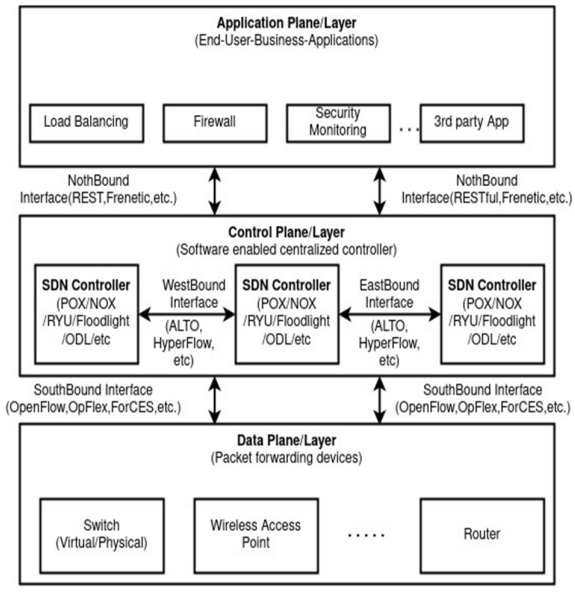

Figure 1 SDN Architecture

2. SDN CONTROLLERS

Controllers in a software-defined network can be defined as NOS (Network Operating System) or the network's brain. NOX, POX[4], Ryu[5], Floodlight[20], Beacon[30], and Open Daylight[31], among others, are open source and commercial SDN controllers. SDN controller utilizes different interfaces in order to interact with other layers in the architecture.

1. Southbound Interface: it can be used to provide interaction between controller and the underlying forwarding devices. There are several protocols available such as OpenFlow[24], OpFlex[25], Forces[26], etc. OpenFlow is a standard protocol supported by the ONF (Open Network Foundation) used as a most common

southbound interface.
2. Northbound Interface: it can be used to provide interaction between controller and the application layer or application plane. Some of the northbound APIs are Frenetic [28], RESTful[29], etc.
3. East-Westbound Interface: can be used to provide interaction among the controllers. Some of the supported East-Westbound interfaces are ALTO[22], Hyperflow[23], etc.

In this paper we have evaluated the performance of python-based Ryu, POX controllers and Java-based Floodlight controller.

2.1. POX Controller

POX is a python-based controller, responsible for providing communication with SDN switches via the OpenFlow or OVSDB protocols[9] . It is inherited from the NOX controller. It is the default controller [4].

2.2. Ryu controller

Ryu [5] is a Japanese term that signifies "flow". NTT labs created Ryu, a Python-based OpenFlow controller. It may be utilized with OpenFlow to gather statistical information from switches. Ryu controller could also be constructed as an traffic monitor, firewall, switch,etc [16, 19]. It includes group of integrated components, like tenant isolation, topology viewer, L2 and L3 switch that can assist in building custom-built network control applications.

2.3. Floodlight controller

Big Switch Networks created Floodlight [20], an open source Java-based SDN controller. The control applications can be simply incorporated with the Floodlight through REST API.

POX, Ryu and Floodlight Controllers Comparison:
The following table shows the feature comparison of three controllers.

Table 1: Feature comparison of controllers

Feature	POX	Ryu	Floodlight
Language Support	Python	Python	Java
OpenFlow Support	V1.0	V1.0 to V1.5, Niciria extensions	V1.0 to V1.5
GUI	Yes	Yes	Yes
REST API	NO	Yes	Yes
OpenSource	Yes	Yes	Yes
Platform support	Linux, Mac, Windows	Linux	Linux
Multithreaded	No	Yes	Yes
Partner	Opensource development	NTT	BigSwitch Network

3. LITERATURE SURVEY

Bholebawa, et.al [9] have evaluated the performance of POX and Floodlight controllers under different network topologies and on two metrics, network throughput and round-trip delay utilizing Mininet emulator. The results obtained shows that Floodlight controller provides minimum delay and maximum throughput compared to POX. Ravindra kumar et.al [7] have assessed the performance of Ryu and Floodlight controllers on several factors such as throughput, jitter, latency, and packet loss across various topologies specifically single, linear, tree, custom, and torus. In most topologies, the experimental assessment revealed that Ryu has higher throughput than Floodlight controller.

Zhao et.al [10] utilized the cbench tool to evaluate the performance of five SDN controllers, including Ryu, NOX, POX, Beacon and Floodlight, on lantency and throughtput metrics. They discovered that the Beacon controller performs better across all criteria.

Karamjeet Kaur et al. [11] have evaluated the performance of POX, Ryu, Pyretic using Iperf tool under RTT and Web Server Latency metrics. They observed that Ryu is best over the other in most cases. C. Fancy e.atl [11], evaluated the performance of Pox and Floodlight controllers respectively on different parameters. With the evaluation they found that Floodlight provides better throughput compared to POX. L. Mamushiane et.al [13] has assessed the performance of Ryu, Floodlight, Open Daylight and ONOS utilizing the Cbench tool. They found that ONOS has better throughput, and Ryu has low latency. Khondoke et. al [14], analyzed the performance of five SDN controllers under several features and criteria namely GUI, programming language, operating system, southbound API, REST API , OpenFlow version, etc. With experimental results they found that results change when priority of criteria is varied, Ryu performance was found to be better than other controllers. Khattak et.al [15] evaluated the performance of two controllers OpenDayLight and Floodlight using Cbench tool. With the evaluation they found that ODL comprise memory leakage problem and Floodlight has average higher response time than ODL. Yanzhen Li et.al [27] evaluated the performance of Ryu and Floodlight controllers using the mininet and qperf tools to compare the bandwidth and latency, the results show that floodlight has higher bandwidth and low latency. Ali et.al [17] have analysed the performance of POX and Ryu under diverge topologies specifically Single, Linear, Tree, etc, under two metrics such as throughput and latency. The evaluation showed that performance of Ryu is better compared to POX. Rastogi et al [18] evaluated the performance of POX and Ryu utilizing Mininet and D-ITG tool. The results obtained demonstrated that performance of POX is better subject to layer 1 switching and performance of Ryu is better subject to layer 2 switching.

4. IMPLEMENTATION ENVIRONMENT

The experiments of this paper are carried out on Windows 10 as the host machine with Intel i5-7th Generation, 12GB RAM and SDN-controller (Ubuntu 20.04 LTS), on which all the three controllers were installed and Mininet (Ubuntu 20.04 LTS) virtual machines installed on VirtualBox. qperf tool was used to assess the bandwidth and latency.

Methodology:

We have implemented 12 different experiments to evaluate the Bandwidth and Latency of POX, Ryu and Floodlight controllers under six distinct topologies with different number of hosts and switches.

4.1 Mininet

Mininet [6] is a network emulation programme. It can be utilized to builds realistic network of virtual hosts, switches, controller and links by employing a simple command on a single system. Each host runs a Linux kernel [6, 8]. Mininet was developed in python and provides extensible API for network customization and experimentation. It provides supports for both CLI and GUI interfaces.

4.2 qperf

qperf (Quick Performance Overview) it is network performance measurement tool that measures bandwidth and latency between two nodes. To perform a test, qperf has to run on one node as a server mode with no arguments and on the other node it has to run as a client mode with arguments.

4.3 Topologies

Topology signifies the layout or arrangements of the nodes or devices in the network. We implemented six topologies in mininet environment namely single, linear, reversed, tree, minimal and custom.

Single topology: it utilizes only one switch to connect all the hosts.

sudo mn –controller = remote,ip = 192.168.56.102 –topo = single, 20

Above command creates a single topology with 20 hosts.

Minimal: minimal topology is the default topology in mininet. It contains one switch and two hosts.

sudo mn –controller = remote, ip = 192.168.56.102

Linear topology: it is identical to bus topology in which the switches are connected in a linear fashion.

sudo mn –controller = remote, ip = 192.168.56.102 –topo = linear, 50

This creates 50 switches in a linear arrangement and connects 50 hosts per switch.

Reversed topology: It is identical to single topology.

Sudo mn –controller = remote, ip = 192.168.56.102 –topo = reversed, 5

It creates a reversed topology consists of 1 switch and 5 hosts.

Tree topology: it uses two parameters namely depth and fan-out. Depth signifies the levels of switches connected and fan-out's signifies the number of output ports that connects switch or hosts.

sudo mn –controller = remote, ip = 192.168.56.102 –topo = tree, 4, 3

Custom topology: is created with 11 switches with 200 hosts.

sudo python3 topo200.py

The table below illustrates the implemented topologies with the number of switches and hosts.

Table 1: Switches and hosts connected under various topologies

Topology	Number of hosts	Number of Switches
Single	20	1
Minimal	2	1
Linear	50	50
Reversed	5	1
Tree	81	40
Custom	200	11

When connecting a topology with a certain controller qperf test was carried out to assess the bandwidth and latency between the two most distant hosts. qperf has run on one node as a server mode with no arguments and on the other node it has to run as a client mode with arguments.

5. RESULTS AND DISCUSSION

The above mentioned topologies were implemented in mininet connecting POX, Ryu and Floodlight controllers one by one. Latency and Bandwidth were compared to assess the performance of the POX, Ryu and Floodlight controllers. With the graph as illustrated in Figure 2, it can be understandable that the bandwidth of Ryu controller is better compared to Floodlight and POX controllers in most of the topologies except single topology. In single topology the performance of POX is better compared to Ryu and Floodlight. The bandwidth of Floodlight and POX controller in minimal topology and the bandwidth of Ryu and POX in reversed topology were almost same.

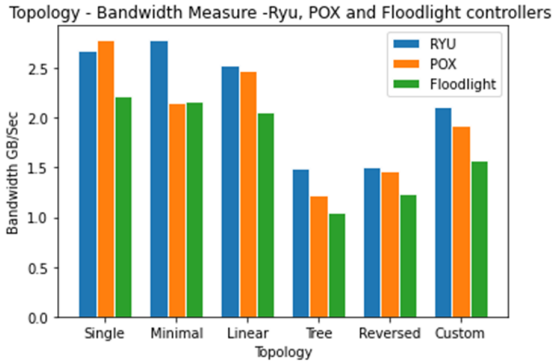

Figure 2 Topology-Bandwidth measures

With the graph as illustrated in Figure 3, it can be understandable that the Latency of Ryu is less than that of Floodlight and POX controllers. POX is competitive with Ryu in most of the topologies. In single topology the latency of POX is slightly better than that of the Ryu.

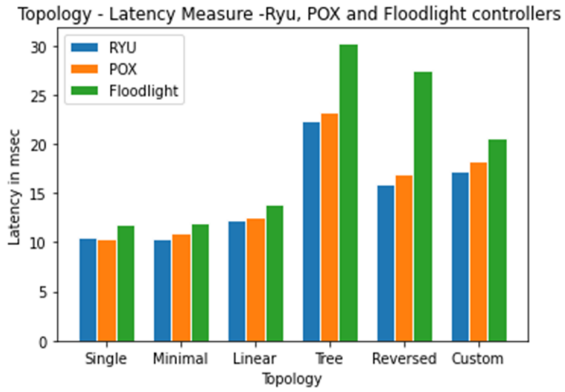

Figure 3 Topology-Latency measures

6. CONCLUSION AND FUTURE SCOPE

With the experimental evaluation we can cease to a decision that Ryu controller outperformed the POX and Floodlight controllers with regard to bandwidth and latency in the majority of scenarios. Except for single, Ryu achieves higher bandwidth in most topologies. It provides less latency in all the topologies. It is noticed that code modification for research purpose is easily achievable with Ryu and POX than that of Floodlight controller. In future we will assess the performance of various other SDN controllers using different network emulation tools and performance metrics.

References

[1.] Arvind T, Dr.K.Radhika," Performance Comparison of POX, Ryu and Floodlight SDN Controllers",AICTE Sponsored International conference on Advances in Computer Engineering and Communication Technology (ICACET-2021), Aditya College of Engineering & Technology,22-23 Oct 2021.

[2.] W. Xia, Y. Wen, C.H. Foh, D. Niyato, H. Xie, "A survey on software-defined networking",IEEE Commun. Surv. Tutor. 17 (1) 27–51, 2014.

[3.] D. Kreutz, F.M. Ramos, P.E. Verissimo, C.E. Rothenberg, S. Azodolmolky, S.Uhlig, "Software-Defined networking: A comprehensive survey", Proc. IEEE103 (1) 14–76, 2014.

[4.] "NOX Repo",Available at: https://github.com/noxrepo/ [Accessed: Sept. 05, 2021].

[5.] "Ryu SDN Framework", Avialable at: https://ryu-sdn.org/ [Accessed: Sept. 05, 2021].

[6.] "Introduction to Mininet",GitHub,available at:https://github.com/mininet/mininet/wiki/Introduction-to-Mininet [Acessed on June 20, 2021]

[7.] C. Ravindra Kumar ,A. Mithilesh,Naresh Kumar N," Performance Comparison of Ryu and Floodlight Controllers in Different SDN Topologies",In proceedings of ICATIECE,IEEE,2019

[8.] "Mininet Overview", Availableat: http://mininet.org/overview/ [Acessed on June 20,2021]

[9.] Bholebawa, I. Z.,Dalal, U. D., "Performance analysis of SDN/OpenFlow controllers: POX versus floodlight", Wireless Personal Communications, 98(2), 1679–1699, 2018.

[10.] Zhao Y, Iannone, L. Riguidel M.."On the performance of SDN controllers: Areality check",IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), 2016.

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

[12.] C. Fancy and M. Pushpalatha,"Performance evaluation of SDN controllers POX and floodlight in mininet emulation environment," International Conference on Intelligent Sustainable Systems (ICISS),pp.695-699,doi:10.1109/ISS1.2017.8389262, 2017.

[13.] L. Mamushiane, A. Lysko and S. Dlamini,"A comparative evaluation of the performance of popular SDN controllers," Wireless Days (WD), pp. 54-59, doi: 10.1109/WD.2018.8361694, 2018.

[14.] R.Khondoker,A.Zaalouk,R.Marx,K.Bayarou,"Feature-based comparison and selection of Software Defined Networking (SDN) controllers", In the proceedings of World Congress Computer Applications and Information Systems (WCCAIS), (pp. 1-7). IEEE, 2014.

[15.] Z K. Khattak, M. Awais, A. Iqbal, "Performance evaluation of OpenDaylight SDN controller", In the proceedings of the 20th IEEE International

Conference on Parallel and Distributed Systems (ICPADS), (pp.671-676). IEEE, 2014.

[16.] J. Ali, S. Lee, and B. H. Roh, "Performance analysis of POX and Ryu with different SDN topologies," ACM International Conference Proceeding Series. pp. 244–249, 2018.

[17.] Jehad Ali, S.Lee, B. Roh, "Performance Analysis of POX and Ryu with Different SDN Topologies", Proceedings of the International Conference on Information Science and System – ICISS '18. pp 244-249, 2018.

[18.] Abhishek Rastogi, A.Bais,"Comparative analysis of software defined networking (SDN) controllers In terms of traffic handling capabilities", In the proceedings of 19th International Multi-Topic Conference (INMIC),(pp. 1-6), IEEE,2016

[19.] Hussein Ali Al-Gboury and Sahar Abdulaziz Al-Talib," Investigate and Compare Software-Defined Network Controllers for UAV Networks Management ", in the proceedings of IOP Conf. Series: Mater.Sci. Eng. 928 022055, 2020.

[20.] "Floodlight Documentation and Installation Guide" : available at https://floodlight.atlassian.net/ wiki/spaces/floodlightcontroller/overview [Acessed on : June 20,2021]

[21.] "RYU SDN Framework Ryubook 1.0 documentation".Available at: https://osrg.github.io/ryu-book/en/html,[Acessed on : June 20,2021] .

[22.] "Opendaylight user guide", Available at: https://docs.opendaylight.org/en/stable-fluorine/user-guide/alto-user-guide.html, [Acessed on : June 22,2021] .

[23.] A. Tootoonchian, Y. Ganjali, "Hyperflow: A distributed control plane for openflow",in the Proceedings of the Internet Network Management Conference on Research on Enterprise Networking, Vol. 3, 2010.

[24.] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J.Rexford, S. Shenker, J.Turner,"OpenFlow: enabling innovation in campus networks",ACM SIGCOMM Comput.Commun. Rev. Vol.38 (2), 69–74, 2008.

[25.] "OpFlex",Available at: https://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/kb/b_Cisco_ACI_and_OpFlex_Connectivity_for_Orchestrators.html ,[Acessed on : Aug 15,2021] .

[26.] A. Doria, J.H. Salim, R. Haas, H.M. Khosravi, W. Wang, L. Dong, R.Gopal, J.M. Halpern,"Forwarding and control element separation (ForCES) protocol specification", RFC 5810 , 1–124, 2010.

[27.] Yanzhen Li, Xiaobo Guo,XuePang, Bo Peng, Xiaoyue Li, P Zhang, "Performance Analysis of Floodlight and Ryu SDN Controllers under Mininet Simulator", ICCC Workshops, DOI:10.1109/ICCCWorkshops49972.2020.9209935, IEEE 2020

[28.] N. Foster, R. Harrison, M.J. Freedman, C. Monsanto, J. Rexford, A. Story, D.Walker, "Frenetic:A network programming language", ACM SIGPLAN Not.46 (9) ,279–291 ,2011.

[29.] "REST API Tuto, rial": available at https://restfulapi.net/ , [Acessed on Auf 15, 2021]

[30.] "The Beacon openflow controller", Available at:http://yuba.stanford.edu/~derickso/docs/hotsdn15-erickson.pdf , [Accessed on, April 28, 2022]

[31.] " Opendaylight", Available at: https://www.opendaylight.org/ , [Acessed on July 22, 2021]

[32.] KS Sahoo, Sagarika M,Mayank T, BK Mishra,B Sahoo, "A Comprehensive Tutorial on Software Defined Network: The Driving Force for the Future Internet Technology", DOI:10.1145/2979779.2983928, AICTC, ACM, 2016.

[33.] Shibo Luo; Jun Wu; Jianhua Li; Bei Pei,"A Defense Mechanism for Distributed Denial of Service Attack in Software-Defined Networks",In the proceedings of Ninth International Conference on Frontier of Computer Science and Technology, DOI:10.1109/FCST.2015.11, IEEE Xplore,2015.

[34.] Md.Tariqul Islam,Nazrul Islam,Md.AI Refat,"Node to Node Performance Evaluation through RYU SDNController", Wireless Personal Communications 112:555–570, DOI:10.1007/s11277-020-07060-4, Springer, 2020.