Performance Analysis of Python Based SDN Controllers over Real Internet Topology

Jisi Chandroth
AI Convergence Networks
Ajou University
Suwon, South Korea
jisichandroth@ajou.ac.kr

Byeong-hee Roh
AI Convergence Networks
Ajou University
Suwon, South Korea
bhroh@ajou.ac.kr

Jehad Ali
AI Convergence Networks
Ajou University
Suwon, South Korea
jehadali@ajou.ac.kr

2022 Thirteenth International Conference on Ubiquitous and Future Networks (ICUFN) | 978-1-6654-8550-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICUFN55119.2022.9829591

***Abstract*—** SDN (Software-Defined Networking) is a revolutionary networking paradigm that separates the data plane and control plane to allow for intelligent network operation. Traffic management, routing, security, and analysis are all responsibilities performed by the control plane. The controller, which offers information to the whole network, is a critical component of the control plane. SDN controllers such as POX, Ryu, Open Daylight, Beacon, and ONOS are frequently utilized. These controllers' behavior is determined by the installed environment, network architecture, and kind of traffic flow, among other factors. It's crucial to compare these controllers in order to learn more about their capabilities. The performance of python-based controllers in Abilene topology is evaluated in this research. Moreover, we also perform a comparison with an increase in the number of nodes. The results of the testing demonstrate that Ryu performs better than POX.

Keywords—POX; RYU; SDN

I. INTRODUCTION

The Fifth Generation (5G) networks and related technologies such as Massive machine-type communications (mMTC), Ultra-reliable low latency communications (uRLLC), Vehicle to vehicle (V2V) infrastructure, etc, are developing rapidly. Due to the rise of these technologies, the number of connected devices is shooting up exponentially and producing a massive amount of data. To provide uninterrupted internet connectivity to all end users and maintain the quality of service is a difficult task [1]. The conventional network architectures are no longer suitable to meet the requirements of today's applications [2]. In order to solve the limitations of traditional networks, the concept of SDN is proposed. In this novel technology, the control plane and data plane are decoupled [3], [4]. Table I shows the comparison between traditional networks and SDN networks.

In the SDN architecture, the network control is separated from forwarding and is directly programmable. This approach enables the real-time control of the underlying network. SDN provides centralized management and sets up new applications devices without impacting the network. In the usage of softwarization, SDN should increase security and lower overall costs. SDN can support both physical and virtual networking.

The multi-layered SDN architecture consists of three planes such as Application plane, Control plane, and Data plane. The Control plane interacts with the Data plane and Application plane through Southbound API and Northbound API respectively. The multilayered SDN architecture is shown in Figure 1. SDN data plane is comprised of the forwarding devices such as switches, routers, and virtual switches. The data plane is responsible for the forwarding of data packets based on the decision made by the controller. The main component in the control plane is the controller which makes intelligent decisions. A wide range of SDN controllers is available in today's market such as POX [5], RYU [6], Floodlight [7], Beacon [8], MUL [9], Onos [10], and

TABLE I.

TRADITIONAL NETWORKS Vs SDN

No	Traditional Networks	SDN
1	Traditional networks offer distributed control over network devices	SDN centrally control the devices using Controllers
2	Traditional networks are non-programmable	SDN networks are programmable. Python, C++, and Java are widely used languages for SDN
3	It has a Closed Interface	SDN has an Open interface
4	Static/manual configuration	Automatic configuration
5	Data and control plane are mounted on the same plane	Data and the control plane are separated from each other. The Control plane acts as a central controller for many data planes.
6	It is difficult to reprogram the existing application	The application can be easily reprogrammed as per the user's need
7	Traditional networks are complex, and the maintenance cost is high due to the replacement of fault hardware	The structural complexity of SDN is low and thus the operating and maintenance costs are low.
8	Due to distributed nature of traditional networks, it is difficult to troubleshoot and report.	It is easy to troubleshoot and report in SDN because it is centrally controlled.

OpenDaylight [11]. The application plane is the place for all SDN applications. SDN applications are software programs written in different programming languages such as Python, Java, or C++ depending on the controller. These programs are directly communicating to the underlying network and collect the desired network behavior to the SDN controller via Northbound Interfaces (NBIs). The applications insist on business applications, user-defined applications, network management, traffic control, etc.

The controllers are consisting the brain of the SDN network, and the performance of the controllers are different in various aspects. Choosing an appropriate controller is always a confusing part of any work. Therefore, an efficient performance analysis is required against these controllers. In this research, a comparative analysis of python based controllers such as POX and RYU has been carried out based on the real internet topology called Abilene Topology. Jitter, Delay, and Throughput are the different performance parameters used in this document.

The rest of the paper consists of VI Sections. Section II discusses the related work. A comparison of python-based controllers is described in Section III. We discuss the simulation model, Network topology, and Performance metrics in Section IV. We discuss our results in Section V. Our conclusion and future works are discussed in Section VI.

II. RELATED WORK

In [12] the performance analysis of Ryu, Open Daylight, Floodlight, and ONOS has been performed against Average delay and throughput. In this paper, the Mininet tool was used for performance evaluation. The papers cover different topologies such as Linear, Tree, and Mesh topologies. The authors perform the comparison against a different number of nodes from 10 to 50. The paper does not include performance measurements such as Jitter, packet loss rate, and Bandwidth.

In [13], the authors compare the python-based Ryu controller against the Java-based Floodlight controller. In this, the Mininet emulator is used for the performance analysis. They were performed to measure the Jitter, packet loss rate, latency, and throughput between hosts, that have maximum distance. The authors found that Ryu ensures better throughput in all conventional topologies.

Work done in [14] is to analyze the performance of ONOS and Floodlight SDN controllers based on TCP and UDP traffic. They have observed traffic transfer, bandwidth, delay, and Jitter in diverse topologies such as Single, Linear, and Tree topologies. The ONOS controller showed better results compared to Floodlight.

The comparison study of seven controllers; POX, RYU, Beacon, Floodlight, Open Daylight, ONOS, and OpenMUL against throughput and latency has been done in [15]. They used Mininet for the simulation tool. From the experiments, the authors found that python-based controllers are not suitable for large networks as they cannot meet high performance and low latency. Java and C-based controllers have better performance and scalability.

The authors of [16] proposed a cloud-based SDN network. In this paper, the authors compare the performance of locally hosted SDN controllers and cloud-based remote controllers. The authors short-listed three SDN controllers such as POX, NOX, and Floodlight based on their popularity and programmable language. Two sets of experiments were conducted for comparison purposes. In Set (A), three different topologies were set up to measure the latency between controllers and Set (B) calculate the throughput of the controllers. The cloud-based controller showed consistent performance compared with locally hosted controllers.

In [17], researchers are comparing widely used seven SDN controllers in the market. The evaluation criteria are jitter, TCP/ UDP Throughput, latency, and the number of end devices. The authors used Mininet as the emulation environment and Iperf, and Gnuplot were used for traffic generation and visualization. While Floodlight shows the better throughput among all other controllers and OpenDayLight shows the better delay.

In this paper [18], the authors evaluate the performance of open-source controllers such as ONOS, Ryu, Floodlight, and OpendayLight. The comparison parameters are latency and throughput. The authors used Cbench, an OpenFlow benchmarking tool for evaluation. From the feature analysis, the authors recommend OpenDayLight because it supports more features in terms of interfaces. From aspects of performance evaluation, ONOS shows the better result in terms of throughput and delay.

In [19-21], the authors conducted a comparison of numerous Mininet controllers. POX and RYU are two of the controllers that have been studied for their performance in various topologies [19], both in terms of throughput and latency. The topologies of real-world networks, on the other hand, are not taken into account in [19]. Because of this, an additional investigation is required.

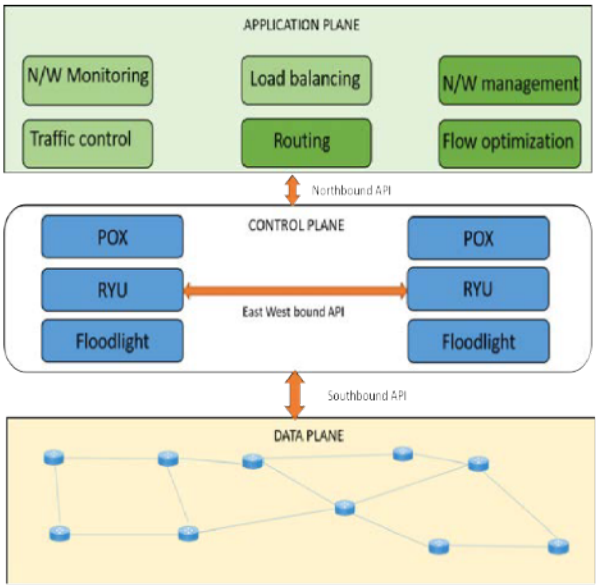

Fig. 1. SDN multi-layered architecture

III. COMPARISON OF PYTHON-BASED SDN CONTROLLERS

A. POX

POX [5] is a python-based Software Defined Networking (SDN) controller. One of the fundamental usages of POX is for developing Open flow control applications due to its faster development and prototyping of new network applications. The POX controller allows an easy way to run Open flow/SDN experiments. POX controller can support hub, switch, load balancer, and firewall devices with Open flow. It is preinstalled with the Mininet emulator. POX contains a number of APIs for developing network control applications. The object called "core" in POX acts as a central point for much of POX's API. Some of the functions it provides are unique but, some are just wrappers around other functionality. One of the major purposes of the core objective is to provide a rendezvous between components. The POX controller enhances the two-way communication between the controller and switches. The communication from the controller to the switch is performed by controller code which sends an open flow message to a particular switch. When the messages are coming from switches are called events-generally an event type corresponding to each message. The salient features of the POX and the Ryu SDN controller are listed in Table 1.

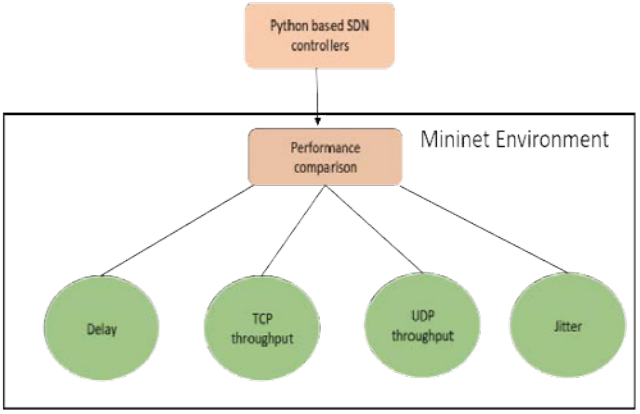

Fig. 2. Simulation Environment

B. RYU

RYU [6] is an open, component-based software-defined networking (SDN) controller developed by NTT. It increases the sharpness of the network by making them easy to manage and adapt to new traffic. RYU provides software components with well-defined APIs so that developers can easily create novel network management and control applications. Organizations can develop customized applications with the help of this component and also developers can instantaneously and efficiently adjust existing components or implement their own components to meet their demands or specific needs. RYU controller supports various protocols such as Open Flow, Netconf, OF-config, etc. for managing network devices. Regarding Open Flow, one of the well-defined and most widely deployed SDN protocols, RYU supports fully 1.0, 1.2, 1.3, 1.4, 1.5, and Nicira Extensions. All the RYU codes are written entirely in Python and all of the codes are freely available under the Apache 2.0 license and open for the public.

TABLE I.

SALIENT FEATURES OF POX AND RYU

Features	RYU	POX
License	NTT communications	ICSI
Source	Apache 2.0	GPL
Language	Python	Python
Target user	Researchers, Developers, Operators	Researchers
Open flow support	V1.0, v1.2,v1.3	V1.0
Open-source	Yes	Yes
GUI	Yes	Python, +QT4
Platform support	Mostly supported on Linux	Linux, Mac OS, Windows
Modularity	Fair	Low
Distributed/Centralized	Centralized	Centralized
Southbound APIs	Open flow 1.0-1.5	Open flow 1.0
Northbound APIs	REST	Ad-hoc
Interface	CLI	CLI, GUI
Multithreading	Yes	No
Documentation	Good	Limited

IV. SIMULATION MODEL

We have compared the performance of two Python-based SDN controllers POX and RYU. The comparison process includes the generation of real internet topology using the Mininet emulator [22]. The simulation environment is shown in Figure 2. The SDN topology with different parameters was run on the launched remote controllers. After that performance tests for Throughput, Jitter and Delay have been conducted on them. Moreover, we also perform an analysis of the two controllers with respect to increasing the number of nodes and evaluating the throughput.

A. Network Topology

A network topology is the graph-based arrangement of the network elements and their links [23]. We have created a real internet topology called Abilene topology [24]. It consists of 11 Open Flow switches and 16 hosts. In the late 1990s, the Internet2 community created this network. It was a high-performance backbone network as shown in Figure 3.

B. Performance Metrics

- Throughput: is the amount of data transmitted successfully from host to destination.
- Delay: is the time required for the packet to be fully received at the destination.
- Jitter: is the difference in delay between two data packets.

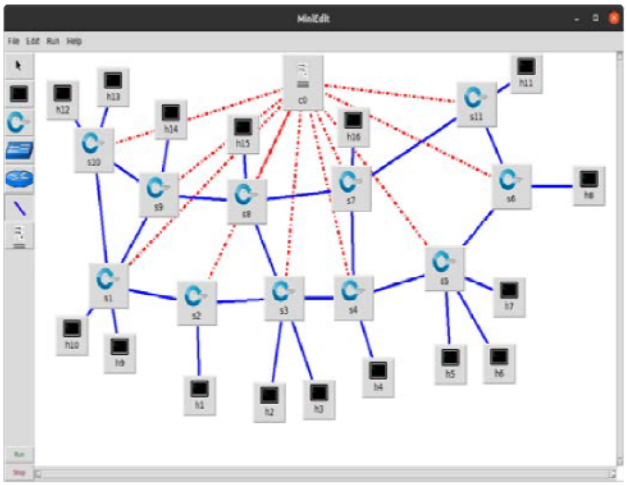

Fig. 3. Abilene Topology

V. RESULTS

In this paper, we created the real internet topology called Abilene using the Mininet emulator and compare the performance of POX and RYU in terms of Delay, Throughput, and Jitter. Iperf test was performed to calculate the TCP and UDP throughput between the host h1 and h16 in the topology. Iperf is a command-line tool designed to test the bandwidth between two network nodes. Iperf can generate TCP and UDP traffic between hosts. Ping command is used for calculating the delay of end hosts. Ping command shows the connectivity between source and destination nodes.

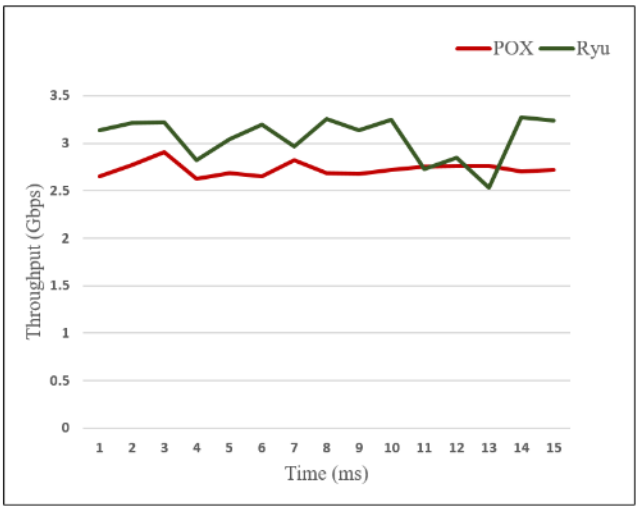

Fig. 4. TCP throughput

Figure 4 represents the TCP throughput comparison of POX and RYU. As shown in the figure, the throughput of the RYU controller is much greater than POX. The average throughput of POX is 2.73 and of Ryu is 3.06 respectively. The UDP throughput comparison of POX and RYU are shown in Figure 5. The throughput of POX and RYU controllers are 10.5 and 12 Gbps/sec respectively. It is clear that the UDP throughput of the Ryu is 87.5% more compared to the POX controller.

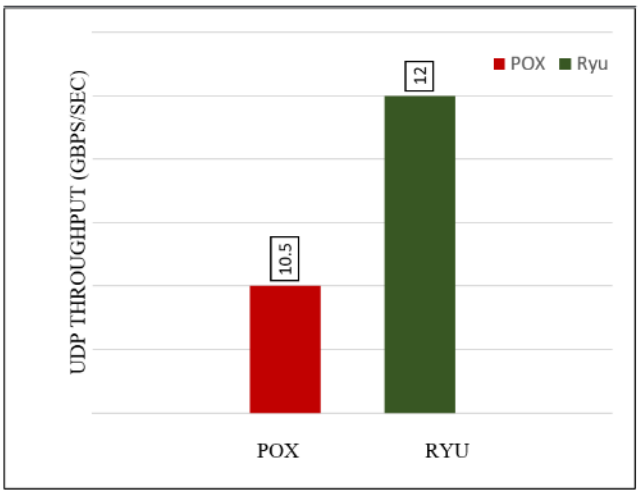

Fig. 5. UDP throughput

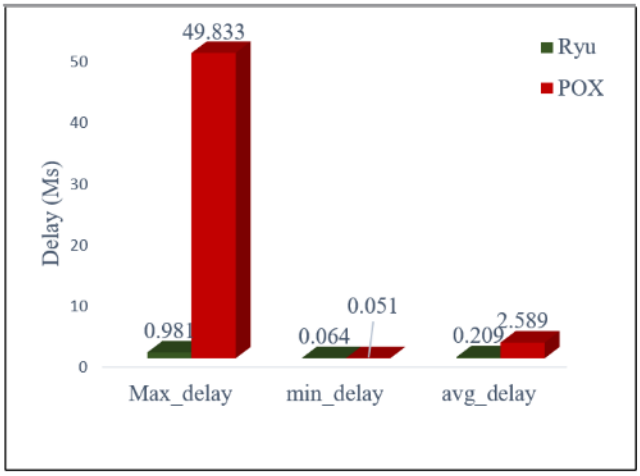

Fig. 6. Delay

Figure 6 shows the comparison of POX and RYU in terms of delay. As shown in the figure, the maximum delay of POX is 49.833 ms, which is far greater than RYU. The minimum delay and average delay of both controllers are almost equal. Figure 7 of the result part describes the performance of the POX and RYU controllers based on the jitter vs time graph where the time interval is 1 millisecond for jitter analysis. The Jitter value of both controllers is increasing with time. It can be noticed that the RYU has the least average Jitter value which is below 0.02 ms.

We also performed an experiment for observing the throughput regarding the number of increases in the number of nodes/switches. Figure 8 shows the comparison of the throughput with an increase in the number of nodes from 1 to 100. We have recorded the throughput in Mininet. Figure 8 shows that RYU surpassed the POX in throughput analysis

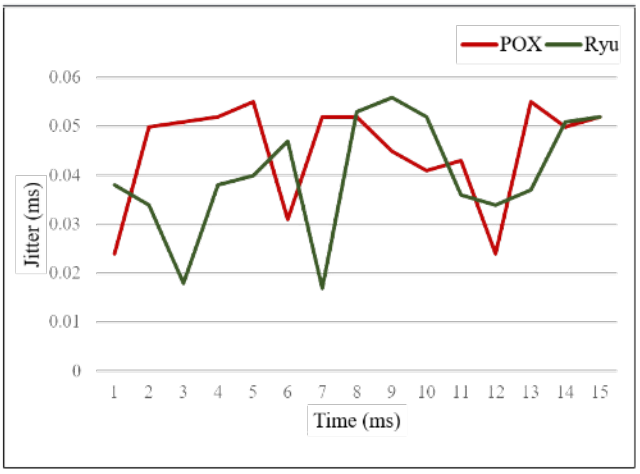

Fig. 7. Jitter

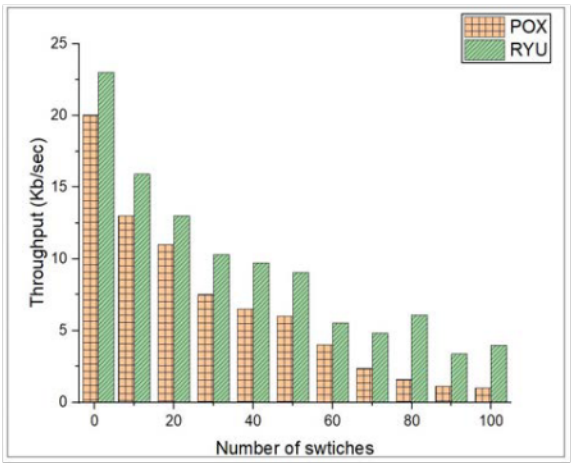

Fig. 8. Throughput comparison with respect to increase in the switches

VI. CONCLUSION

In this paper, we have performed a comparative analysis of the two-python based SDN controllers i.e. POX and RYU. First, we investigated several features of these two controllers. Then, we created Abilene topology, which is a real internet topology using the Mininet emulator. Finally, we compared the performance of POX and RYU SDN controllers using the above-mentioned topology. The performance parameters are Delay, Throughput, and Jitter. The results show that the decrease in delay and increase in throughput for Abilene topology while using in RYU is high. More and more, we have also performed a comparison of the two controllers with an increase in the number of nodes which showed that RYU outperforms the POX. In the future, we will investigate more controllers in several real-Internet topologies. Moreover, we will consider more parameters for analysis.

VII. ACKNOWLEDGMENT

This work was supported partially by the BK21 FOUR program of the National Research Foundation of Korea funded by the Ministry of Education(NRF5199991514504) and by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program(IITP-2022-2018-0-01431) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

REFERENCES

[1] T. O. Olwal, K. Djouani and A. M. Kurien, "A Survey of Resource Management Toward 5G Radio Access Networks," in IEEE Communications Surveys & Tutorials, vol. 18, no. 3, pp. 1656-1686, thirdquarter 2016, doi: 10.1109/COMST.2016.2550765.

[2] L. Chettri and R. Bera, "A Comprehensive Survey on Internet of Things (IoT) Toward 5G Wireless Systems," in IEEE Internet of Things Journal, vol. 7, no. 1, pp. 16-32, Jan. 2020, doi: 10.1109/JIOT.2019.2948888.

[3] Z. Zaidi, V. Friderikos, Z. Yousaf, S. Fletcher, M. Dohler and H. Aghvami, "Will SDN Be Part of 5G?," in IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 3220-3258, Fourthquarter 2018, doi: 10.1109/COMST.2018.2836315.

[4] F. Bannour, S. Souihi and A. Mellouk, "Distributed SDN Control: Survey, Taxonomy, and Challenges," in IEEE Communications Surveys & Tutorials, vol. 20, no. 1, pp. 333-354, Firstquarter 2018, doi: 10.1109/COMST.2017.2782482.

[5] https://noxrepo.github.io/pox-doc/html/

[6] https://S. Asadollahi, B. Goswami and M. Sameer, "Ryu controller's scalability experiment on software defined networks," 2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC), 2018, pp. 1-5, doi: 10.1109/ICCTAC.2018.8370397. scalability experiment on software defined networks," 2018 IEEE International

[7] https://www.sdxcentral.com/networking/sdn/definitions/what-isfloodlight-controller/ What Is a Floodlight Controller? September 16, 2014 (ICCTAC), 2018, pp. 1-5, doi: 10.1109/ICCTAC.2018.8370397.

[8] https://openflow.stanford.edu/display/Beacon/Home.html

[9] https://www.sdxcentral.com/directory/kulcloud-inc ltd/openmulcontroller/ OpenMUL SDN Controller

[10] https://opennetworking.org/onos/

[11] https://www.sdxcentral.com/networking/sdn/definitions/opendaylightcontroller/ What Is an OpenDaylight Controller? AKA: OpenDaylight Platform= Erin Moriarty-SilerSeptember 16, 2014

[12] A. K. Arahunashi, S. Neethu and H. V. Ravish Aradhya, "Performance Analysis of Various SDN Controllers in Mininet Emulator," 2019 4th International Conference on Recent Trends on Electronics, Information, Communication Technology (RTEICT), 2019, pp. 752-756

[13] R. K. Chouhan, M. Atulkar and N. K. Nagwani, "Performance Comparison of Ryu and Floodlight Controllers in Different SDN Topologies," 2019 1st International Conference on Advanced Technologies in Intelligent Control, Environment, Computing Communication Engineering (ICATIECE), 2019, pp. 188-191, doi: 10.1109/ICATIECE45860.2019.9063806.

[14] A. H. Eljack, A. H. M. Hassan and H. H. Elamin, "Performance Analysis of ONOS and Floodlight SDN

Controllers based on TCP and UDP Traffic," 2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), 2019, pp. 1-6, doi: 10.1109/ICCCEEE46830.2019.9071189.

[15] A. Shirvar and B. Goswami, "Performance Comparison of Software-Defined Network Controllers," 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), 2021, pp. 1-13, doi: 10.1109/ICAECT49130.2021.9392559.

[16] K. Basu, M. Younas, A. W. Wan Tow and F. Ball, "Performance Comparison of a SDN Network between Cloud-Based and Locally Hosted SDN Controllers," 2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService), 2018, pp. 49-55, doi: 10.1109/BigDataService.2018.00016.

[17] D. Lunagariya and B. Goswami, "A Comparative Performance Analysis of Stellar SDN Controllers using Emulators," 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), 2021, pp. 1-9, doi: 10.1109/ICAECT49130.2021.9392391.

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

[19] Ali, J., Lee, S. and Roh, B.H., 2018, April. Performance analysis of POX and Ryu with different SDN topologies. In *Proceedings of the 2018 international conference on information science and system* (pp. 244-249).

[20] Ali, J., Roh, B.H. and Lee, S., 2019. QoS improvement with an optimum controller selection for software-defined networks. *Plos one*, *14*(5), p.e0217631.

[21] Ali, J. and Roh, B.H., 2021. Quality of service improvement with optimal software-defined networking controller and control plane clustering. *Comput. Mater. Contin*, *67*, pp.849-875.

[22] De Oliveira RLS, Shinoda AA, Schweitzer CM, Prete LR. Using Mininet for emulation and prototyping software-defined networks. in Proc. IEEE Colombian Conf. on Commun. and Computing (COLCOM); 2014; 1–6.

[23] Lu, YH., Leu, FY. (2020). Dynamic Routing and Bandwidth Provision Based on Reinforcement Learning in SDN Networks. In: Barolli, L., Amato, F., Moscato, F., Enokido, T., Takizawa, M. (eds) Advanced Information Networking and Applications. AINA 2020. Advances in Intelligent Systems and Computing, vol 1151. Springer, Cham. https://doi.org/10.1007/978-3-030-44041-1_1

[24] M. Bloem, T. Alpcan, S. Schmidt and T. Basar, "Malware Filtering for Network Security Using Weighted Optimality Measures," 2007 IEEE International Conference on Control Applications, 2007, pp. 295-300, doi: 10.1109/CCA.2007.4389246.