International Journal of Advances in
Soft Computing and Intelligent Systems (IJASCIS)
2024, Vol 03, Issue 02, 303-314
ISSN: 3048-4987

Original Paper

PERFORMANCE EVALUATION OF SDN CONTROLLERS: ANALYSING THE TCP TRAFFIC MANAGEMENT IN POX, RYU, AND ODL

Shanu Bhardwaj[a,*], Shailender Kumar[b]
Ashish Girdhar[c]

[a,b]Department of Computer Science and Engineering, Delhi Technological University, Delhi, India, shanubhardwaj1@gmail.com
[c]Department of Computer Science and Applications, Kurukshetra University, Kurukshetra, India,

ABSTRACT

Software-defined networking (SDN) is a revolutionary networking paradigm that separates the data and the control plane. The controller is one of SDN's leading entities that controls the information flow in the network. Therefore, the research deals with a thorough performance differentiation of three prominent SDN controllers: POX, Ryu, and OpenDaylight (ODL). The study aims to evaluate these controllers' effectiveness in controlling the network traffic by focusing on performance parameters such as Transmission Control Protocol (TCP) mean, packet loss, and jitter. The experimental setup employed Mininet, a network emulator, to create a consistent virtual network environment for all controllers. Each controller was tested in isolated virtual machines, ensuring controlled and unbiased results.

The experimental results reveal distinct performance differences among the controllers. In the research experimentations, the highest TCP mean throughput and superior performance among all controllers are achieved by ODL consistently, and minimum loss of the data packets and jitter is observed across all-time instances for high-demand, large-scale networks. This study shows that choosing the right SDN controller is crucial as it depends on particular network requirements to guide network administrators and researchers when choosing the SDN controller best for their network.

KEYWORDS

Software-defined networking, SDN controllers, Traffic analysis, TCP traffic management

1. INTRODUCTION

SDN is an amazing network methodology that separates the control and physical planes. In this view, it merges control and dynamic setup. Tight coupling of control and data planes in individual devices leads to traditional networks' frequent rigidity and complexity [1]. These restrictions are overcome by decoupling these network planes, allowing incorporated network knowledge, administering delegations, and increasing adaptability. This centralized architecture will give us a global view of the network, as shown in Fig. 1. Thus, it facilitates deploying new services and applications with reduced time, enhances performance, and maximizes resource utilization [2].

Ryu, POX, and ODL are the most extensively utilized regulators out of the many SDN regulators accessible. Each controller presents novel aspects and capabilities handling various use cases and needs. It is essential to understand the distinctions and how they can be used to select the most appropriate controller for a particular system management need [3]. Among

others, Ryu is a well-known open-source SDN controller that is well-praised for its simplicity, adaptability, and user-friendliness [4]. It supports multiple protocols, such as Open Flow, the SDN standard definition of control plane to data plane communication. POX is another open-source SDN controller used for SDN examination and experimentation. The idea was to be lightweight and easy to use. Hence, it is an excellent choice for learning and exploring SDN concepts. Finally, the Linux Foundation funded the development of ODL, an open-source, scalable SDN controller. It aims to expedite the adoption of SDN and NFV with a cooperative and simple path toward improvements. On top of this, traffic examination is essential for SDN controllers because numerous factors counter the compelling administration and evolution of organizational efficiency [5].

SDN controllers offer a single point of view of the whole network, allowing continuous intrusion and traffic (information) flow observation. This allows centralized control network administrators to optimize traffic flows dynamically, identify congestion points, spot anomalies [6], etc. This allows SDN controllers to program the organization through SDN controllers, allowing robotized traffic designing and improving the organization's proficiency and execution. Additionally, SDN supports the execution of cutting-edge safety efforts by allowing fine-grained command over traffic streams and fast reaction to anticipated risks. Thus, SDN controllers are a powerful tool for modern traffic analysis because of these capabilities, which can enhance end-user quality of service, reduce operational costs, and increase network reliability [7]. Used in lots of different contexts to improve network agility, security, and efficiency, these controllers will be put to work. Ryu can be found in networks of any size, small to medium, primarily because of its ease of use and ability to develop quickly. Habitually, POX is employed in educational establishments and analysis laboratories to instruct and investigate ideas of SDN. As it is scalable and offers comprehensive features, ODL is preferred over other alternatives in large-scale deployment in telecommunications and enterprise environments [8]. It becomes essential to choose a proper controller, as each has advantages and a distinct capacity to suit a particular niche application.

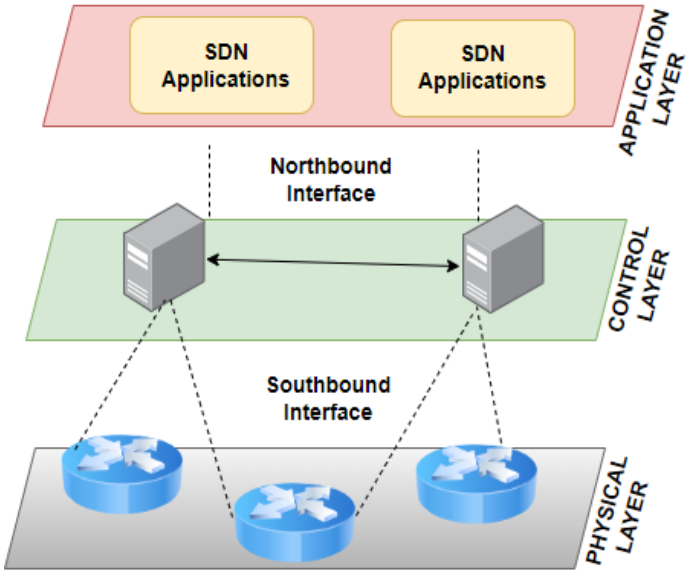


Figure 1. SDN Architecture

The author compares Ryu, POX, and ODL SDN controllers in this paper. Several performance parameters have been used for this comparison, such as control response time, data flow efficiency, and network latency. Based on the simulation results, the author concludes which SDN controller is the best for traffic analysis. This study highlights the strengths and

weaknesses of each controller based on different performance metrics, providing valuable insights into their effectiveness and suitability for network management and optimization.

2. BACKGROUND

This section highlights the SDN controllers' distinct development histories and primary use cases.

2.1. Traditional Networking vs SDN

Due to the decentralized idea of connection plan-making in conventional networks, in some cases, traffic analysis is usually constrained by it as control and information planes are executed inside gadgets. Network managers find it tedious and likely to be riddled with irregularities [9] to gather information from several sources. This is often not a powerful tool to analyze traffic patterns or diagnose issues because of the local visibility and slow response times that it entails. However, traffic analysis is done centrally through SDNs with software-based controllers accessing the entire network. This centralization allows for the detection of anomalies, improvement of performance, enforcement of security policies, and the enabling of real-time traffic flow monitoring. Additionally, the programmability found in SDNs enables the use of the latest analytics and machine learning algorithms to gain greater insight and control by traffic management administrators [10]. The advantage of SDNs is that traffic analysis is more accurate and efficient, making the response time to network conditions and potential threats quicker and more efficient.

Table 1. Properties of the SDN controllers.

Features	**RYU**	**POX**	**ODL**
Language Support	Python	Python	Java
Platform Support	Linux, Windows	Linux	Linux
Structure	Lightweight	Lightweight	Extensible
Virtualization	Mininet	Mininet	Mininet
Northbound API	REST etc.	RPC etc.	REST etc.
Southbound API	OpenFlow, BGP, etc.	OpenFlow	OpenFlow, NETCONF, etc.
Documentation	Good	Basic	Extensive
Scalability	Small scale	Small scale	Large scale
Modularity	Basic	Basic	High

2.2. Ryu controller

Ryu is a well-known open-source SDN controller known for being straightforward, adaptable, and simple. OpenFlow, the standard for SDN communication between the control and data planes, is one of the protocols that it supports, as depicted in Fig. 2. Ryu gives a far-reaching set of libraries and instruments that work with the fast turn of events and sending of organization applications [11]. It is broadly utilized in scholarly examination and tiny to medium-sized creation conditions because of its direct engineering and broad documentation. Some of Ryu's real-time applications include traffic engineering, network automation, and security monitoring, making it an adaptable option for various network scenarios.

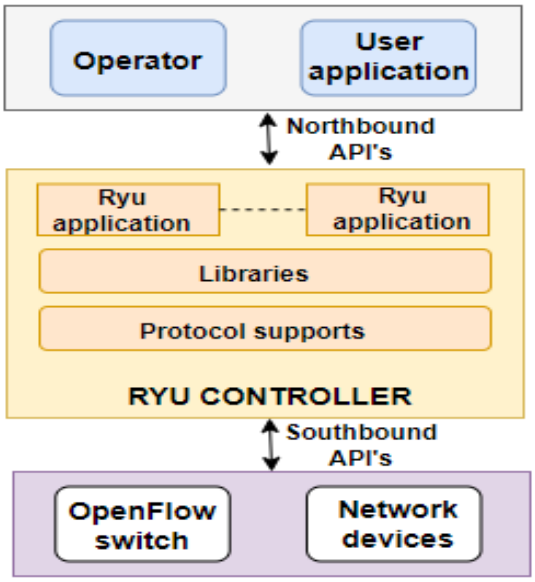


Figure 2. Architecture of Ryu SDN controller

2.3. POX controller

POX is another open-source SDN controller instrumental in SDN examination and training [12]. It is intended to be lightweight and straightforward, making it a superb decision for learning and exploring different avenues regarding SDN ideas, as shown in Fig. 3. POX's simplicity makes it a valuable tool for developing and evaluating new SDN applications, even though it may lack the same level of sophistication and scalability as Ryu or ODL, also shown in table 1. POX is frequently utilized in experimental setups to validate novel networking concepts and in academic settings to teach SDN principles in real-world scenarios [13].

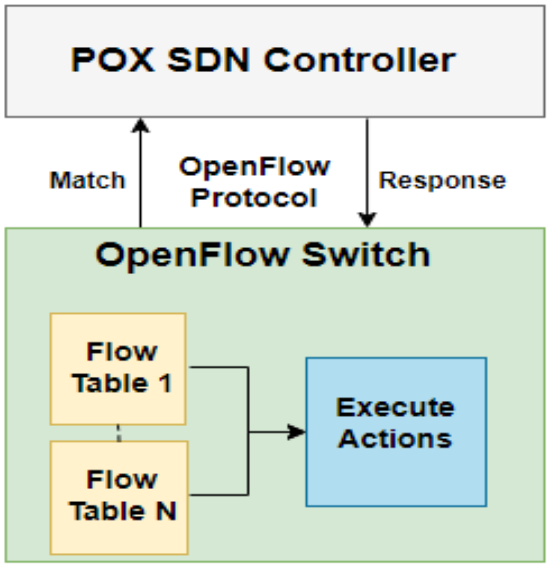


Figure 3. The architecture of the POX SDN controller

2.4. ODL controller

ODL is a vigorous and versatile open-source SDN controller created under the Linux Establishment. A development process that is open and collaborative aims to accelerate the use

of SDN and NFV. OpenFlow, NETCONF, and BGP are just a few of the many southbound protocols that ODL can handle, making it ideal for large-scale production environments. Its modular design makes it easy to customize and integrate with various network management tools [14]. ODL is broadly utilized in broadcast communications, server farms, and undertaking networks for organization virtualization, administration coordination, and high-level network analytics [15]. Fig. 4 depicts the architecture of the ODL controller in the SDN environment.

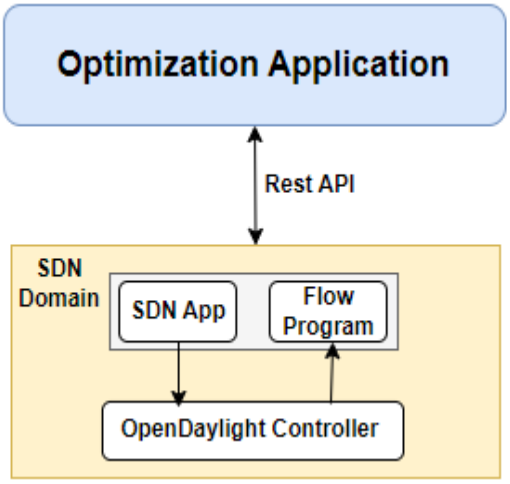


Figure 4. The architecture of the ODL SDN controller

3. METHODOLOGY

To implement Ryu, POX, and ODL controllers for traffic analysis in an SDN environment, the first step involves ensuring that the servers or virtual machines have adequate resources and a compatible operating system, such as Ubuntu. Designing an appropriate network topology is crucial for practical traffic analysis, and this has been achieved using network emulation tools like Mininet to create virtual networks suitable for testing. The flow of the implementation of the proposed research work is depicted in Fig. 5 and Table 2.

The next step is the installation of the SDN controllers. For the Ryu controller, dependencies such as Python3 and related packages have been installed. Ryu itself can be installed using Python's package installer (pip). Similarly, cloning the POX repository and installing the POX controller requires Python 2.7 and pip. Beginning POX is direct with a basic order to start the controller. ODL, being more complex, includes downloading the ODL conveyance from the authority site, extracting the documents, and beginning the ODL utilizing the Karaf compartment. This setup guarantees that each controller is prepared for design and combination.

Each controller's traffic monitoring applications or components must be set up when the controllers are configured for traffic analysis. For Ryu, existing applications, such as simple_monitor_13.py, can be utilized, and this has begun through the Ryu manager. Regarding POX, traffic examination parts can be incorporated into the POX environment by beginning POX with these particular parts. ODL requires establishing elements like **old-l2switch-switch**, which work with traffic investigation. Sending traffic-checking applications inside ODL includes successfully utilizing the Karaf climate to deal with these highlights.

Coordinating the controllers with the network is the last step, where devices like Mininet assume a vital part. Installing Mininet and constructing a network topology capable of

communicating with the SDN controllers is essential. Each controller must be configured for this network to be managed and monitored. For instance, Mininet can begin with a predefined geography that interfaces with the SDN controllers, empowering them to accumulate traffic information and perform examinations. This integration considers ongoing traffic observing, anomaly detection, and execution improvement across the network, utilizing the abilities of Ryu, POX, and ODL.

Table 2. Steps of the implementation.

Steps to be followed	**Implementation**
Setting up the environment	• Hardware and software requirements • Network topology
Installing SDN controllers	• Ryu controller i. Install dependencies ii. Install Ryu • POX controller i. Install dependencies ii. Install POX • ODLcontroller i. Download ODL distribution ii. Extract the file tar –xvf distribution-karaf-x.x.x.tar.gz iii. Start ODL cd distribution-karaf-x.x.x./bin/karaf
Configuring the controllers for traffic analysis	• Ryu controller i. Develop or use existing Ryu applications for traffic analysis. ii. Start application ryu-managerpath/to/your/application.py • POX controller i. Create or use existing POX components for traffic analysis. ii. Start POX with the traffic analysis. • ODL controller i. Install necessary features for traffic analysis. ii. Develop or use existing ODL applications for traffic monitoring. iii. Deploy the application in the ODL environment using Karaf.
Integrating the controllers with the network	• Mininet setup i. Installation ii. Network Topology
Traffic analysis	• Data collection • Real-time monitoring

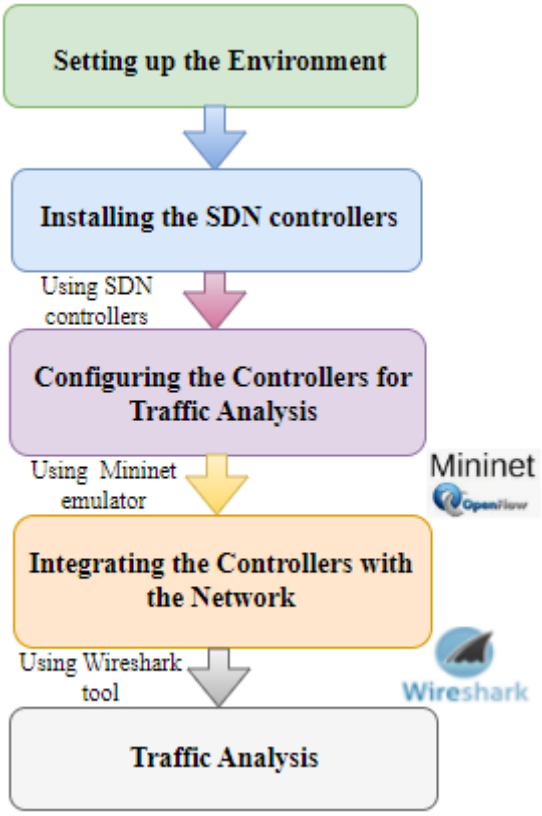


Figure 5. Implementation flow of the proposed SDN setup.

4. Results and Discussion

This research's results and discussion section include a comparative analysis of three prominent SDN controllers, Ryu, POX, and ODL, as well as an analysis of their implementation and ability to perform traffic analysis. This begins with successfully installing and configuring each Mininet virtual network environment controller and then evaluates the controllers' traffic monitoring capabilities. The practical implications of each controller are examined based on key performance metrics such as latency, scalability, ease of integration, and others.

4.1. TCP mean

In TCP mean [16], we used a methodical approach that included environment setup, traffic generation, and mean estimation to precisely examine how well the POX, Ryu, and ODL controllers handled TCP traffic within an SDN. This method ensures we have methodically evaluated and differentiated each controller's traffic analysis capabilities.

The TCP was used to measure POX, Ryu, and ODL for evaluation at 30 seconds, 20 seconds, and 10 seconds, respectively, as shown in Fig. 6. The TCP mean is compared to various time instances in the table. For each time interval, the performance of the controllers is recorded. At the 30-second mark, ODL shows the highest throughput, followed by Ryu. At 20 seconds, ODL's throughput decreases, while Ryu and POX register 26.6 and 25.05, respectively. At 10 seconds, ODL maintains a high throughput of 32.49, Ryu peaks at 29.4, and POX records 23.6. These results suggest that ODL consistently outperforms the other controllers in terms of mean TCP throughput across all time intervals. At the same time, Ryu and POX show variable performance, with Ryu generally performing better than POX.

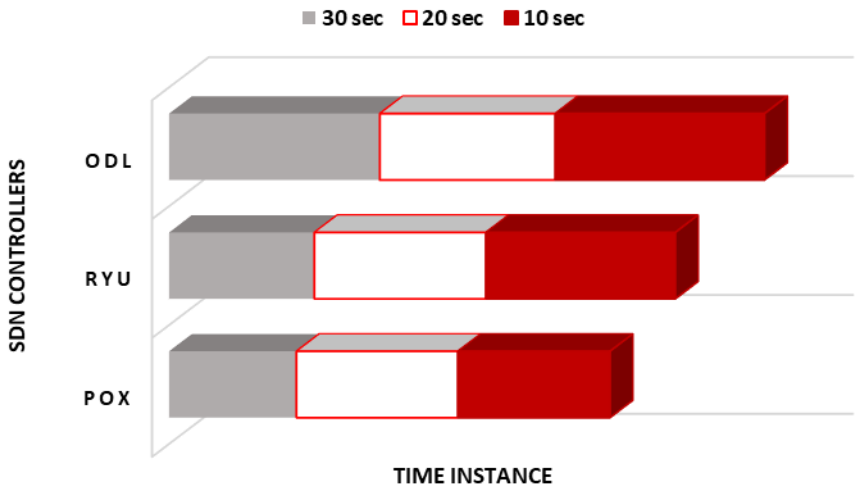


Figure 6. TCP means of POX, Ryu, and ODL controller.

4.2. Jitter and Packet loss

The bar graph illustrates packet loss percentages and jitter values (in milliseconds) for three SDN controllers, POX, Ryu, and ODL, at three different time intervals: 10 seconds, 20 seconds, and 30 seconds, as shown in Fig. 7.

At 10 seconds, POX exhibits the highest packet loss, followed by Ryu with a noticeable amount and ODL with minimal packet loss. POX shows the highest jitter, while Ryu and ODL have significantly lower jitter values. At 20 seconds, Packet Loss: POX continues to have the highest packet loss, with Ryu showing moderate packet loss and ODL maintaining a low packet loss rate. Once again, POX has the highest jitter, while Ryu and ODL show very low jitter. At 30 seconds, POX remains with the highest packet loss, Ryu has moderate packet loss, and ODL displays the least packet loss. POX still has the highest jitter, with Ryu showing less and ODL having minimal jitter.

Fig. 8 provides an overview of network performance metrics, specifically jitter and packet loss, across different network controllers named POX, Ryu, and ODL at various bandwidths such as 500 Mbps, 600 Mbps, 700 Mbps, and 800 Mbps. Jitter measures the variability in packet arrival times, while packet loss represents the percentage of lost data packets. Among the controllers, Ryu demonstrates the best performance overall, with the lowest jitter and packet loss across most bandwidths, particularly excelling at 700 Mbps with minimal packet loss. ODL also shows strong performance but slightly higher jitter than Ryu, especially at higher bandwidths. While consistent in jitter, POX exhibits higher packet loss than Ryu and ODL. This indicates that Ryu is the most efficient in terms of both consistency and reliability of data transmission. At the same time, ODL performs well but with a bit more variability, and POX shows some limitations in maintaining low packet loss.

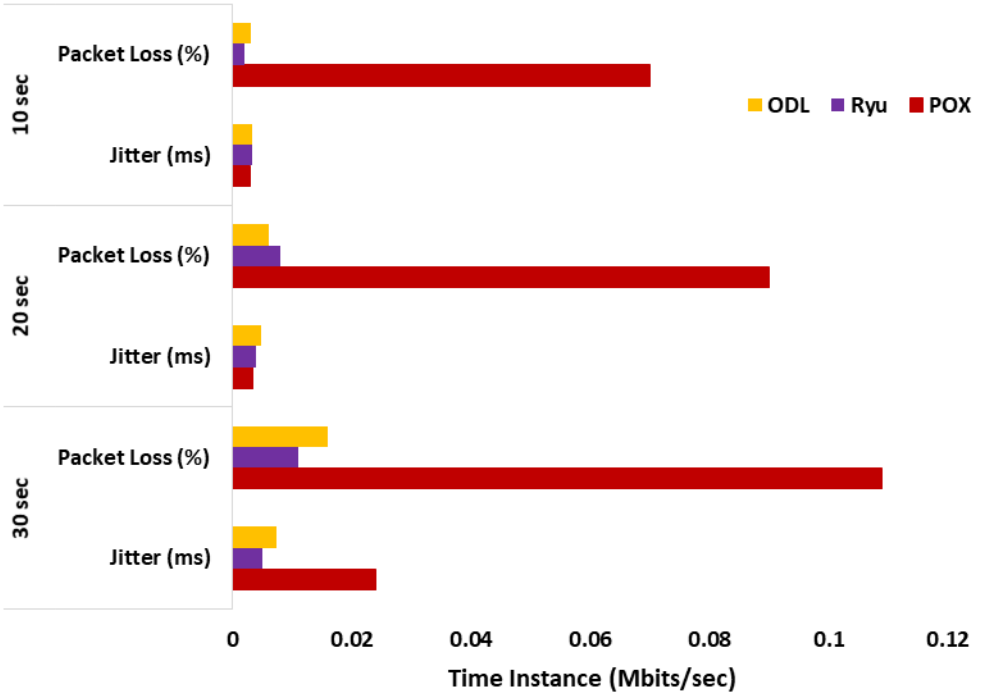


Figure 7. Jitter and packet loss of the SDN controllers in distinct time instances.

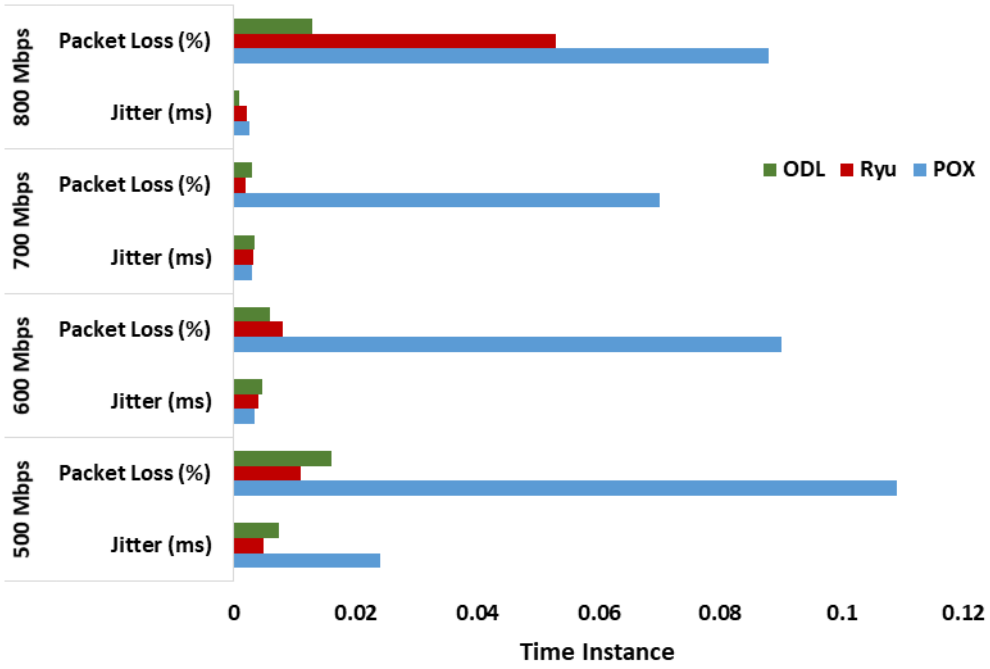


Figure 8. Jitter and packet loss of the SDN controllers in distinct bandwidth.

5. CONCLUSION

The performance comparison of POX, Ryu, and ODL SDN controllers for TCP throughput, packet loss, and jitter was made, and variation was observed. ODL performed better than the others, as ODL achieved the highest mean of the TCP throughput, the least amount of packet loss, and lower jitter than the other flows, making it efficient in large-scale high-performance networks. Ryu demonstrated better results than the previous experiments with an average interconnect time of 63.069 milliseconds and packet transfer rate of 7.050 Mbps, thus exceeding the POX performance but still lacking ODL's capabilities, implying that the Ryu is fine for medium-sized Networks with the comprehension of easier use while still being potentially faster. However, due to its lowest throughput, highest packet loss rate, and highest jitter, POX is ideal for small-scale networks and educational and testing environments where consolidation is valued most. Thus, this research emphasizes the importance of selecting the right SDN controller to provide high-quality and reliable network performance and adapt to specific network needs and environments.

Disclosure of Interests. The author has no competing interests to declare relevant to this article's content.

REFERENCES.

[1] Chahal, J. K., Bhandari, A., & Behal, S. (2024). DDoS attacks & defense mechanisms in SDN-enabled cloud: Taxonomy, review, and research challenges. Computer Science Review, 53, 100644.

[2] Abdi, A. H., Audah, L., Salh, A., Alhartomi, M. A., Rasheed, H., Ahmed, S., & Tahir, A. (2024). Security Control and Data Planes of SDN: A Comprehensive Review of Traditional, AI and MTD Approaches to Security Solutions. IEEE Access.

[3] Bhardwaj, S., & Girdhar, A. (2023). Network traffic analysis in software-defined networking using Ryu controller. Wireless Personal Communications, 132(3), 1797-1818.

[4] Bhardwaj, S., & Girdhar, A. (2021, November). Software-defined networking: A traffic engineering approach. In 2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON) (pp. 1-5). IEEE.

[5] Adanza, D., Gifre, L., Alemany, P., Fernández-Palacios, J. P., González-de-Dios, O., Muñoz, R., & Vilalta, R. (2024). Enabling traffic forecasting with cloud-native SDN controller in transport networks. Computer Networks, 250, 110565.

[6] Shirmarz, A., & Ghaffari, A. (2020). Performance issues and solutions in SDN-based data center: a survey. The Journal of Supercomputing, 76(10), 7545-7593.

[7] Bhardwaj, S., Panda, S. N., & Datta, P. (2020, December). Layer-Based Attacks in the Ternary Planes of Software-Defined Networking. In 2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE) (pp. 292-295). IEEE.

[8] Maleh, Y., Qasmaoui, Y., El Gholami, K., Sadqi, Y., & Mounir, S. (2023). A comprehensive survey on SDN security: threats, mitigations, and future directions. Journal of Reliable Intelligent Environments, 9(2), 201-239.

[9] Indrason, N., & Saha, G. (2024). Exploring Blockchain-driven security in SDN-based IoT networks. Journal of Network and Computer Applications, 103838.

[10] Jiang, W., Han, H., He, M., & Gu, W. (2024). ML-based pre-deployment SDN performance prediction with neural network boosting regression. Expert Systems with Applications, 241, 122774.

[11] Bhardwaj, S., & Panda, S. N. (2022). Performance evaluation using RYU SDN controller in software-defined networking environment. Wireless Personal Communications, 122(1), 701-723.

[12] Cabarkapa, D., & Rancic, D. (2021). Performance Analysis of Ryu-POX Controller in Different Tree-Based SDN Topologies. Advances in Electrical & Computer Engineering, 21(3).

[13] Rizaldi, M. I., Yusuf, E. A. S., Akbi, D. R., & Suharso, W. (2024). A Comparison of Ryu and Pox Controllers: A Parallel Implementation. Jurnal Online Informatika, 9(1), 1-9.

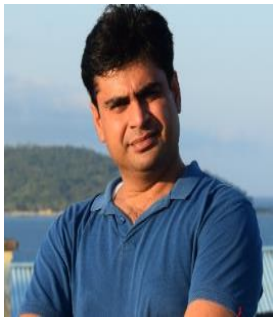
[14] Hassen, H., & Meherzi, S. (2024). Performance evaluation of centralized and distributed controllers in software-defined networks. International Journal of Wireless and Mobile Computing, 27(2), 103-117.

[15] Singh, A., Kaur, N., & Kaur, H. (2022). Extensive performance analysis of OpenDayLight (ODL) and open network operating system (ONOS) SDN controllers. Microprocessors and Microsystems, 95, 104715.

[16] Adhikari, T., Kumar Khan, A., & Kule, M. (2024). ProDetect: A Proactive Detection Approach of the TCP SYN Flooding Attack in the SDN Controller. IETE Journal of Education, 1-12.

Authors

Shanu Bhardwaj is currently pursuing PhD in the Department of Computer Science & Engineering at Delhi Technological University, Delhi, India. She has completed her B. Tech in Computer Science and Engineering at Kurukshetra University, Kurukshetra (India), and her M.E. at Chitkara University, Punjab (India). Her research areas are Software-Defined Networking (SDN), the Internet of Things (IoT), Wireless Sensor Networks (WSN), and Network Security. She completed her "Application of Big Data in Construction Law" internship at Tamkang University, Taiwan.



Prof. Shailender Kumar has worked as a Professor in the Department of Computer Science & Engineering at Delhi Technological University since December 2018. He has a total of more than 17 years of Teaching Experience. He has 20 SCI/SCIE and other publications. His primary research areas are Network Security, Computer Networks, Database Management Systems, and Machine Learning.

Dr. Ashish Girdhar has worked as an Assistant Professor in the Department of Computer Science & Applications at Kurukshetra University since 2022. He has a total of more than 12 years of Teaching

Experience. He has a PhD in Image Processing from Thapar Institute of Engineering and Technology, Patiala. He has 7 SCI/SCIE Publications and a few other publications. His primary areas of research are Image Processing and Machine Learning.