Indian Journal of Information Sources and Services
ISSN: 2231-6094 (P) Vol.14, No.1, 2024, pp.39-50

DOI: https://doi.org/10.51983/ijiss-2024.14.1.3926

Analysis of Software-Defined Networking (SDN) Performance in Wired and Wireless Networks Across Various Topologies, Including Single, Linear, and Tree Structures

Anil Ram[1] and Swarnendu Kumar Chakraborty[2]
[1]Research Scholar, [2]Associate Professor,
Department of Computer Science and Engineering, National Institution of Technology, Arunachal Pradesh, India
E-mail: anilram.nitap@gmail.com, swarnendu@nitap.ac.in
(Received 13 December 2023; Revised 27 January 2024, Accepted 15 February 2024; Available online 26 February 2024)

***Abstract* - The increasing prominence of the internet and the resulting heightened demand for flexibility and agility have rendered traditional networking solutions inadequate for meeting current computing needs. Software-Defined Networking (SDN) emerges as a solution to achieve these goals. A controller plays a crucial role in determining the success of SDN. Therefore, it is necessary to assess and compare the various SDN controllers used across different industries. In this study, we evaluate the effectiveness of two recognized SDN controllers, POX and Ryu. Our research employs the Mininet-Wi-Fi emulator, and we assess the aforementioned controllers using metrics such as Jitter, throughput, packet loss, and delay, utilizing the Distributed Internet Traffic Generator (D-ITG). What sets our research apart is its examination of network performance across both wired and wireless transmission modalities. Fast Ethernet was chosen as the speed for the wired medium, as it had not been studied before. Additionally, the packet size ranged from 128 to 1,024 bytes. We used single, linear, and tree topologies for comparison. Our experimental findings demonstrate that, in the majority of cases, Ryu offers significantly reduced latency, packet loss, and jitter compared to POX. Furthermore, the Ryu controller outperforms POX in terms of throughput, particularly in wireless networks.**
***Keywords:* Software-Defined Networking, Ryu, POX, Delay, Jitter, Bitrate, Packet Loss**

I. INTRODUCTION

Many individuals are concerned about the current hardware-based network infrastructure because data-forwarding equipment, such as routers and switches, are frequently loaded with control requirements and rules. This issue arises from the fact that conventional networks are now not only overly difficult to construct and maintain, but also resistant to the new service revolution (Tivig, 2021). In a typical network, these proprietary and heterogeneous forwarding devices are tightly packed alongside the data planes, which manage data forwarding, and the control Plane (CP), which act as the intelligences of the networks (Ma, 2022).

The node must be configured and the flow data pathways must be programmed by the control plane. The control statistics is utilized to determine data plane (DP) forwarding at the hardware level once these paths have been designed and transmitted down to the data plane (Kazi, 2021). The usual network system uses a dispersed method of network administration because there is no CP abstraction of the whole network. Thus, networks have become extra complicated and challenging to monitor and constitute when something drives wrong. The idea of SDN has been put out as a remedy for the issues.

Network management is significantly simplified by the SDN construction, which enables centrally measured aptitudes and a comprehensive view of the complete physical network. This unified entity permits here and now control of all the essential devices and bids customizable control of the whole network (Cherian, 2021).

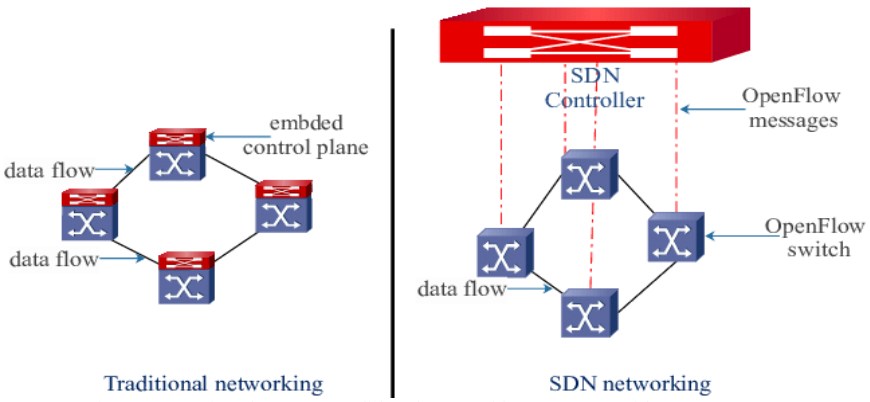

Fig. 1 Comparison between Traditional Networking Structure with SDN Structure

A conventional network and an SDN network's functional planes differ in Figure 1 (Maaloul, 2018). Even though the CP manages Flow Control (FC) and upholds a centralized controller's global viewpoint of the network; SDN preserves the DP logic within the network fundamentals.

Switches and other physical layer network components are given commands by the controller. This specific controller will be in charge of providing programmable interfaces and forwarding decision-making capabilities, allows user-written programs to bring about the function of the network devices in accordance with a set of high-level rules. These regulations include network address translation, load balancing, switching, firewalling, and routing (Askar, 2021).

The controller's performance determines a substantial portion of the SDN's performance. Nowadays, consumers have a huge selection of controllers to pick from, both free and paid. For this reason, a thorough evaluation technique is needed to choose the best controller for each state based on the routing protocols, topology, workload, and Quality of Service (QoS) needs, and all of which take a big influence on the SDN controller's efficiency. Packet loss, Delay, throughput, and jitter are a few eminent metrics that can be castoff to gauge the effectiveness of the recognized SDN controller (Nóvoa, 2021). In this study, the performance requirements for Ryu and POX controllers under various network topologies and workloads will be compared. Southbound interface (SBI) instructions, such as OpenFlow, are used by the SDN controller to communicate with the data plane switches.

In 2008, the first protocol to separate the DP and the CP was OpenFlow. OpenFlow, the most popular uniform SBI follows the core SDN fundamental of isolating the DP from the CP. It outlines the network modifications and data plane device communication that the controller should do. OpenFlow switches have mostly replaced traditional switches because they are less difficult to manage and program because they are vendor-specific.

II. REVIEW OF LITERATURE

Salman *et al.,* analysed the effectiveness of various controllers in wireless networks as a result of the good influence of SDN on altering traditional network principles. Using the Mininet-Wi-Fi emulator, they compared the controllers for floodlights, Ryu, POX, open network operating system (ONOS), and other systems. A network with four hosts and four access points associated in a linear architecture was used for the evaluation. According to the authors, of the four SDN controllers, floodlight has the best jitter and delay performance. Additionally, they demonstrated that Ryu and ONOS had the poorest jitter and delay performances, but Python Network Operating System (POX) had an average performance across the board (Salman, 2022).

Yusof *et al.,* calculated SDN's delay and jitter compared to traditional networks. Author found that SDN average jitter and total delay are 3 times less per packet, resulting in greater network efficiency under different traffic states. According to their findings, SDN improves network effectiveness by reducing the network bugs caused by repetitive attempts to transfer between CP and DP every time a packet arrives. The distributed CP architecture (DCP) is also recommended. However, this architecture has its own challenges, such as the way controllers are positioned to achieve the best results (Yusof, 2021).

Islam *et al.,* looked into the RYU SDN controller's performance on a wired network. To determine the round-trip time (RTT) of a basic single basic topology made up of five users, the authors employed the Ping and iperf programs. Their research investigated different point to point performance under the transmission and user datagram protocols (Islam, 2020).

The features of conventional networks and SDN were compared by Mohammadi, R., *et al.,* Based on throughput, packet loss and delay, they evaluated the POX controller's performance in five topologies. They measured performance measures using Wireshark. They came to the conclusion that the tree topology is the worst and the linear topology is the finest in terms of delay and throughput (Mohammadi, R., 2021).

On the other hand, Koulouras, I., *et al.,* assessed Ryu's presentation in a tree topology with three fans and two depths (i.e., 4 switches and 9 hosts). They analysed Ryu's performance using a selection of metrics, including delay, bandwidth, jitter, and packet loss. Ping, iperf, and Wireshark were some of the measurement tools they employed. Additionally, they recommended that the study be expanded to include various SDN controllers and network topologies, such as linear, single, and ring (Koulouras, I., 2022).

In various network topologies, Mamushiane, L., *et al.,* matched the performance of the SDN ONOS and ODL controllers. The effectiveness of these controllers was examined in linear, tree, and single topologies. Based on the D-ITG tool, a comparison was done. They demonstrated that ODL had extremely low performance, whereas the ONOS controller had the greatest results across all scenarios and metrics (Mamushiane, L., 2021).

Finally, Keerthana, B. *et al.,* assessed the RYU SDN controller's routine in a wired based set-up. To determine the throughput and latency of a linear network topology with a configurable number of the switches (2 switches, 4 switches, 8 switches, 16 switches, 32 switches, and 64 switches), the author used the Cbench and Wireshark programs. Additionally, he suggested that the research be expanded to assess additional performance indicators including as jitter, packet loss, and round-trip time. In both wired and wireless networks, we compare POX and Ryu controller performance across different topologies through various workloads (Keerthana, B. 2022).

TABLE I COMPARISON BETWEEN AUTHOR'S CONTRIBUTION

Author's	Wired Network				Wireless Network			
	Delay	Jitter	Packet Dropped	Bitrate	Delay	Jitter	Packet Dropped	Bitrate
Salman, M. I. *et al.,*	No	No	No	No	Yes	Yes	No	No
Yusof, K. M., *et al.,*	Yes	Yes	No	No	No	No	No	No
Islam, M. T., *et al.,*	Yes	No	No	No	Yes	No	No	No
Mohammadi, R., *et al.,*	Yes	No	Yes	No	No	No	No	No
Koulouras, I., *et al.,*	Yes	Yes	Yes	No	No	No	No	No
Mamushiane, L., *et al.,*	Yes	No	Yes	Yes	Yes	No	Yes	Yes
Keerthana, B. *et al.,*	Yes	Yes	Yes	No	Yes	Yes	Yes	No
Proposed scheme	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

III. THEORETICAL BACKGROUND

A. Software Defined Networking (SDN)

In comparison to conventional network topologies, the SDN architecture has a variety of benefits, such as the ability to adjust programmability, traffic engineering policy, great efficiency, and a centralized view of the network (Singh, 2022). SDN is currently creating a scalable and dynamic architecture to dramatically advance future networks (Koulouras, 2022) (Ramdhani, 2021).

Because of its unique architecture, the SDN is independent of any registered hardware or software. It offers, the network's programmability, centralized management, less agility, more flexibility, operating costs and lower capital, neutrality, and simple modification. It is being adopted by academia and major IT firms as a result of its fame (Umar, 2021) (Cabarkapa, 2021). Additionally, network change is simpler to implement and less prone to error. Dynamically responding to variations in the process of developing network servers, the network state, applications, and services is made simpler. Three levels make up the SDN network's architecture: the Application Layer (AL), the Control Layer (CL), and the Infrastructure (data) Layer (DL) (Tivig, 2021) (Ma, 2022) (Koulouras, 2022) (Ramdhani, 2021) (Lucas, 2021) (Aldabbas, 2021) (Kelian, 2023).

Application layer: This layer houses a variety of user-specific programs. It covers network tools and services used often by businesses, like load balancing, firewalls, and security programs (Aldabbas, 2021) (Bhardwaj, 2022).

Control layer: A brainly centralized controller, which is located at this layer, is the primary element of the SDN design. The function of this layer is to control how network devices behave generally. It is capable of sending commands to any device on the network and has thorough knowledge of every one of them. Through Northbound APIs, the control layer receives instructions from the AL and sends them to the DL. Additionally, it retrieves data from the DL and relays it back to the AL, including bugs, host tracks, and statistics (Lucas, 2021) (Prabakaran, 2021) (Mohammed, 2023).

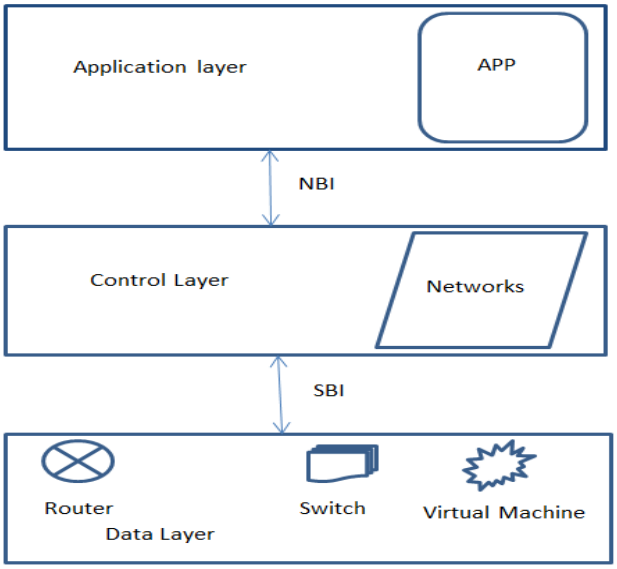

Fig. 2 SDN Architecture

All networking components that are involved in packet forwarding on the network are included in the data layer. This layer's devices don't have any decision-making logic; instead, they simply carry out the commands from the control layer on the packets that are sent to them. The southbound APIs handle communication between the CL and DL. Utilized for communication is OpenFlow (Saputra, 2021) (Prabakaran, 2021) (Balarezo, 2021) (Mohammed, 2023).

B. SDN Controllers

Network designs are being deployed using a number of open-source SDN controllers, including ONIX, Kandoo, OVS, Ryu, POX, floodlight, ONOS, ODL, and OpenDayLight, Trema, Faucet, Beacon, NOX, NodeFlow (Febrianto, 2021). Fig. 3 (Tseng, 2018) depicts the SDN controller's architectural layout. The northbound interface (NBI) and southbound interface (SBI) modules, together with a few potential controller-using applications, are shown in the diagram. The SBI API is used to establish connections in between DP and CP (Balarezo, 2021) (Khorsandroo, 2021). This API is also known as OpenFlow or a substitute in other SDN systems in the case of an Open

SDN controller. The statistics and device tracking and discovery, the topology, flow management, and device administration, are the controller's main responsibilities. A group of internal controller modules carry out all of these functions (Mishra, 2021).

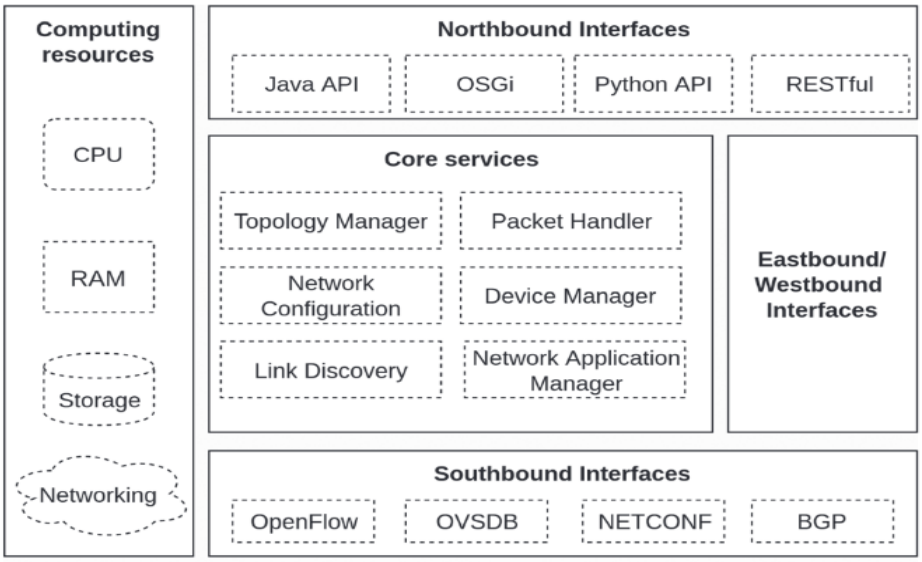

Fig. 3 SDN Controller's Architecture

Given the significance of the role of an SDN controller, a brief explanation of packet forwarding and routing, the controller's two main functions, is warranted. If the current flow in the table does not contain any information about host N, the first switch in the SDN environment collects the very first data flow from the host M and delivers to its SDN controller when node M and node N begin talking (Mishra, 2021). Based on its applications and services, the controller then completes the packet-in and encapsulates it to produce a packet-out. This will be sent back to the switch along with every other switch between the source and the destination. Each switch FT has all the data required to make the optimum routing option. The performance of numerous SDN controllers in various scenarios has to be compared. Some of the most prominent SDN controllers are included in Table II, along with the programming languages and the platforms they serve that were employed in their development (Numan, 2019) (Koulouras, 2022).

TABLE II COMPARISON BETWEEN SDN CONTROLLERS

	RYU	**POX**	**OpenDaylight**	**Floodlight**	**Trema**
Open Source	**Yes**	**Yes**	**Yes**	**Yes**	**yes**
Interfaces	SB (OpenFlow), SB Management (OVSDB, JSON)	SB OpenFlow	SB (OpenFlow), NB (REST & Java RPC)	SB OpenFlow, NB (REST & Java)	SB OpenFlow
GUI	Yes	Yes	Yes	Yes	No
Virtualization	Mininet & OpenvSwitch	Mininet & OpenvSwitch	Mininet & OpenvSwitch	Mininet & OpenvSwitch	Built-in Emulation Virtual Tool
Transport Layer Security support	Yes	Yes	Yes	Yes	Yes
OF support	OF v1.0 v2.0 v3.0 & Nicira Extensions	OF v1.0	OF v1.0	OF v1.0	OF v1.0
OpenStack Networking (quantum)	No	Strong	Medium	Medium	Weak
Rest API	No	Yes	Yes	Yes	No
Documentation	Poor	Medium	Medium	Good	Medium
Productivity	Moderate	Moderate	Moderate	Moderate	High
Platform support	Mac, Linux, and Windows	Linux, Virtual Machine	Linux	Mac, Linux, and Windows	Linux
Modularity	Medium	Medium	High	High	Medium
Language support	Python	Python	Java	Java + any language that uses REST	C/Ruby

C. POX

POX is an OpenFlow-compatible, Python-based open-source SDN emulator or controller for building SDN technology. A POX SDN controller can also create load balancers, firewalls, switches, and OpenFlow devices. When the OpenFlow protocol is present, the forwarding devices are directly controlled and accessed by the POX controller. It's quick and easy, which makes it perfect for research, experiments, and demonstrations (Salman, 2022) (Latif, 2022). The foundation of POX controller is the understanding that every SDN network operation and device is a separate component that can be utilized at anytime and anyplace. It must handle all forms of interaction between apps and SDN devices (Salman, 2022).

D. RYU

Developed solely in Python, the Ryu controller is a component-based and open-source, SDN system (Keerthana, 2022). It offers an event-driven user interface design style in which the program's flow is determined by events and practices the OF protocol to connect with the switches to alter by what means the network manages traffic movements. Event classes that refer to messages received from associated switches are exported by the module ryu.controller.ofp_event. Control applications and SDN network management are made simple to implement by Ryu's software components with well-defined APIs. The designed network can also be seen in the GUI. A collection of important Ryu components for SDN applications includes OpenFlow representational API (OFREST), Firewall, OpenStack, and Quantum (Ali, 2023). The primary objectives of these applications are to collect network intelligence through the use of a controller, run algorithms for analytics, and use the controller to compose new rules. Furthermore, Ryu supports several protocols, such as OpenFlow, OF-config, and Netconf (RFC 6241), for network infrastructure management. The Ryu and all OpenFlow versions (1.0 to 1.5) work together flawlessly (Salman, 2022) (Keerthana, B. 2022) (Askar, 2021).

IV. METHODOLOGY

This research framework aims to assess the effectiveness of SDNs, in particular for Single, Linear, and Tree Topologies to W-N network and W-L network. To optimize network design and deployment, this framework aims to provide insight into the performance and management of these topologies. Under different scenarios, measure the performance metrics like delay, bitrate, jitter, and packet dropped. To identify strengths and weaknesses, compare the performance of single, linear, or tree topologies.

The simulation environment will be described in this section. There will also be demonstrations of single, linear, and tree network topologies. Finally, the procedure for evaluating and contrasting four separate characteristics (packet loss, jitter, delay, and bitrate) will be explained.

A. Simulation Environment

A single personal computer (PC) with an RYZEN 5 2.90 GHz CPU, 4 cores, 6 logical processors, and 8 GB of RAM was used to build the test environment detailed in this section. The computer had installed VMware Workstation Pro, and Ubuntu was used to build the virtual machine. 1 core, 6 GB of RAM, and 40 GB space were used to build the virtual machine. A Mininet-Wi-Fi emulator with Ryu and POX controller implementations is included in the VM. Because Mininet-Wi-Fi is an open source and offers simple modelling of connections, nodes, all network components, and controllers, it is widely utilized. On the other hand, because both Ryu and POX employ Python as their programming language, those controllers were chosen.

B. Topology

Building an SDN topology using hosts, switches, and a Mininet-Wi-Fi emulator is part of the evaluation process. The performances of three topologies are examined. Figure shows the topologies chosen for Wired Network (W-N) and Wireless Networks (WL-N), including single, linear, and tree topologies.

Single: Only one switch is utilized in a single topology, and all hosts are connected to it. Figures 4(a) and 4(b) depict the emulated single 12-host network.

Linear: Each OpenFlow-capable switch in a linear topology is connected in a straight line, much like in bus architecture. Figures 4(c) and 4(d) show an illustrative linear network with eight switches and ten hosts.

Tree: The concepts "depth" and "fan-out" determine the topology of a tree. Fan-out represents the numeral of output ports that hosts or switches will connect to, whereas depth describes the number of switch levels. A simulated tree network with a fan-out and depth of 2 (15 switches and 16 hosts) is shown in Figures 4(e) and 4(f).

Each node has been given a distinct MAC address and an IP address from the address space (10.0.0.0/24). The IP/MAC addresses for node h1 and node h12 are (10.0.0.1/00:00:00:00:00:01 and (10.0.0.12/00:00:00:00:00:12, respectively).

The OpenFlow switches and access points were situated 100 meters away from the hosts. Fast Ethernet was selected for the wired connection, while 802.11g was selected for the wireless. The virtual IP address (127.0.0.1) was castoff to make access points and switches link to the POX controller or Ryu. Access points and Switches that implement OF v1.0 are the devices used to transfer traffic flow from one host to another. Because the POX controller only supports this version of Openflow and in the interest of fair comparison, this version was utilized.

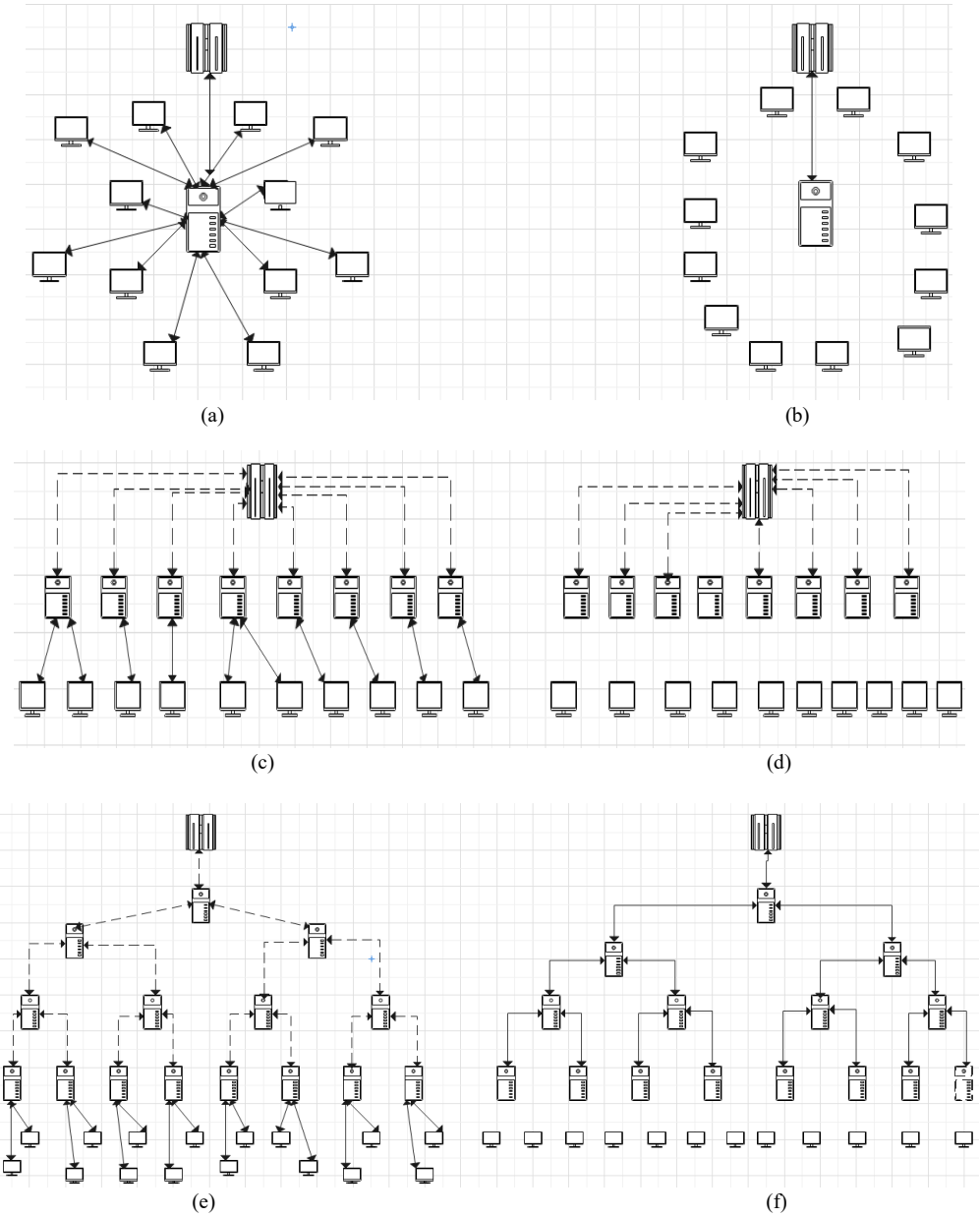

Fig. 4 (a) Single W-N topology (b) Single WL-N topology (c) Linear W-N topology (d) Linear WL-N topology (e) Tree W-N topology (f) Tree WL-N topology

C. Metrics

When the network architecture was put into practice, DITG instructions were used to calculate key network presentation factors such as latency, packet loss, bitrate, jitter, and jitter. Table III described the parameters used in this research work.

TABLE III DESCRIPTIONS OF SIMULATION PARAMETERS

Parameter Name	Description
Tools used	D-ITG, Mininet, Mininet-WiFi
Network type	Wired and Wireless
Network packet size	Wired- 10^6 pks/s, Wireless- 10^6 pk/s
Protocol	TCP, UDP
/ITGSend -T UDP -m rttm -a 10.0.0.8 -c 128 -C 1000008 -t 10008	D-ITG command line to investigate the performance.
Controller Used	RYU
API used	NBI, SBI, OpenFlow

The initial and end nodes in each Mininet-WiFi topology will be selected and set using the standard terminal emulator Xterm after each topology has been completed. Host Y acts as the server in both trials, while Host X acts as the client. Host X is configured using the D-ITG command lines that is ". /ITGSend -T UDP -m rttm -a 10.0.0.8 -c 128 -C 1000008 -t 10008" to start the performance investigation. While the ". /ITGRecv" command is being used to setup host Y.

For both controllers, each procedure was run twice in independent test sets. The W-N's constant rate was set to 10^6 packets per second (pks/s), while the WL-N's constant rate was also set to 10^5 pks/s.

The UDP protocol was used to measure packet drop, jitter, and delay. On the other hand, TCP was used in place of UDP when the D-ITG command was used to measure bitrate. In each stage, we perform tests ten times. The final result is the average of the ten outcomes. Additionally, we gradually raise the packet size by a multiple of 2 from 128 to 1,024. Table IV contrasts the studies reviewed in section 2 with our research technique.

TABLE IV DIFFERENT METHODOLOGY COMPARISON

Reference	Topology	Controller	Tools	Performance
Salman, M. I.	Linear	RYU, POX, Floodlight, ONOS	Ping, iperf	Throughput, Jitter, Delay
Numan, P. E., *et al.,*	Single	POX	Ping, iperf	RTT, Jitter, Delay
Islam, M. T., *et al.,*	Single	Ryu	Ping, iperf	Jitter, Delay
Mohammadi, R., *et al.,*	Single, Linear	POX	Wireshark	Throughput, Packet loss, Delay
Keerthana, B., *et al.,*	Tree	Ryu	Wireshark, Ping, iperf	Throughput, Jitter, Delay, Packet loss
Mamushiane, L., *et al.,*	Single, Linear, Tree	ONOS, ODL	D-ITG	Throughput, Jitter, Delay, Packet loss
Koulouras, I., *et al.,*	Single, Linear	Ryu	Cbench, Wireshark	Throughput, Delay
Our research	Single, Linear, Tree	Ryu, POX	D-ITG, Mininet	Throughput, Jitter, Delay, Packet loss, Bitrate

V. RESULTS AND DISCUSSION

A. Average delay

According to Figure 5, Ryu outperforms POX in the three topologies of wired networks, particularly in the linear and tree topologies. In comparison to POX, where the latency often reduces in tree and linear topologies, furthermore it demonstrated that the delay in Ryu grows as the packet size increases. Figure 6 demonstrates that in wireless networks, both controllers achieve about identical performance, with the poorest for 512 bytes and the greatest result occurring for packet sizes of 128 bytes. SDN performs better in small-size packet applications in our settings. Additionally, the POX controller performs better with tree topology than linear.

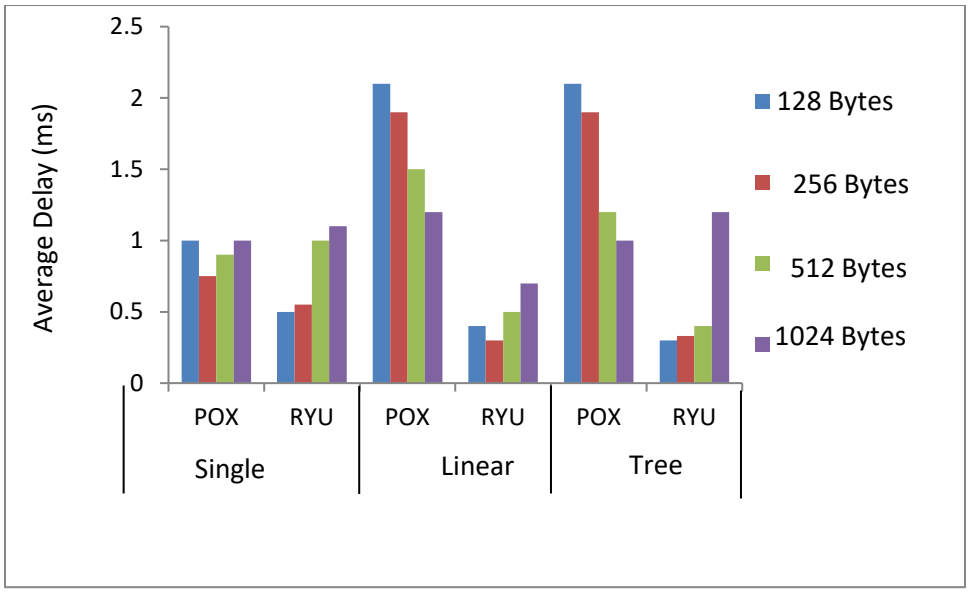

Fig. 5 Average Delay in W-N

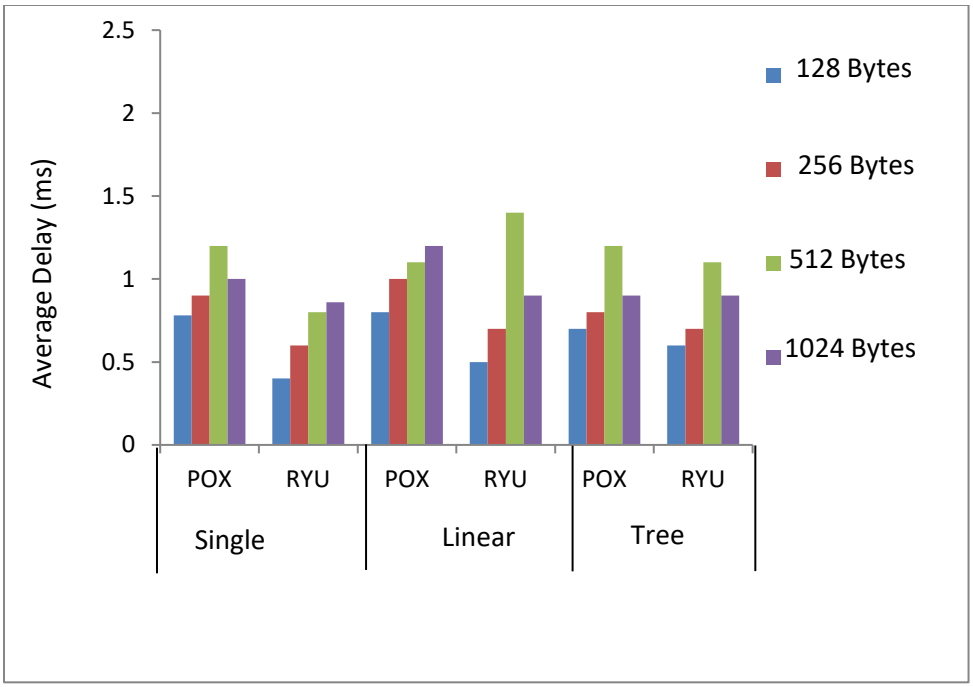

Fig. 6 Average Delay in WL-N

B. Average Jitter

Figure 7 demonstrates that RYU has decreased jitter as the packet size increases although the jitter is amplified by a POX SDN controller, but it's specifically once packet size drops. Both controllers' jitter exhibits delay-like behavior. As seen in Figure 8, the average jitter in wireless networks is the same for both controllers. Additionally, with a POX controller, tree topology performs better than linear. 128 bytes was shown to be the ideal packet size for tenders that require less jitter, whereas 1,026 bytes is the worst.

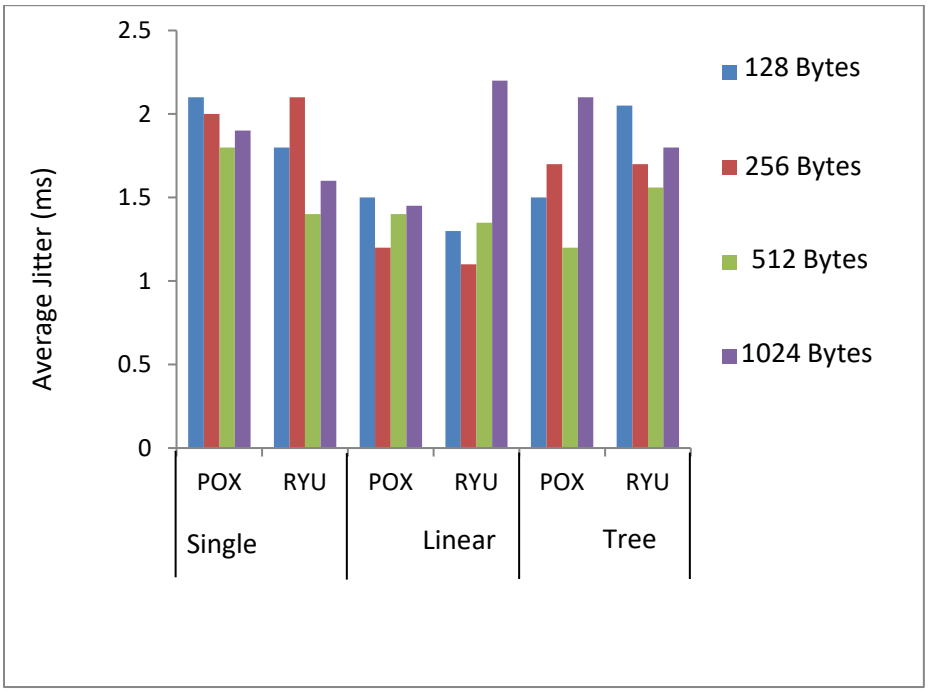

Fig. 7 Average Jitter in W-N

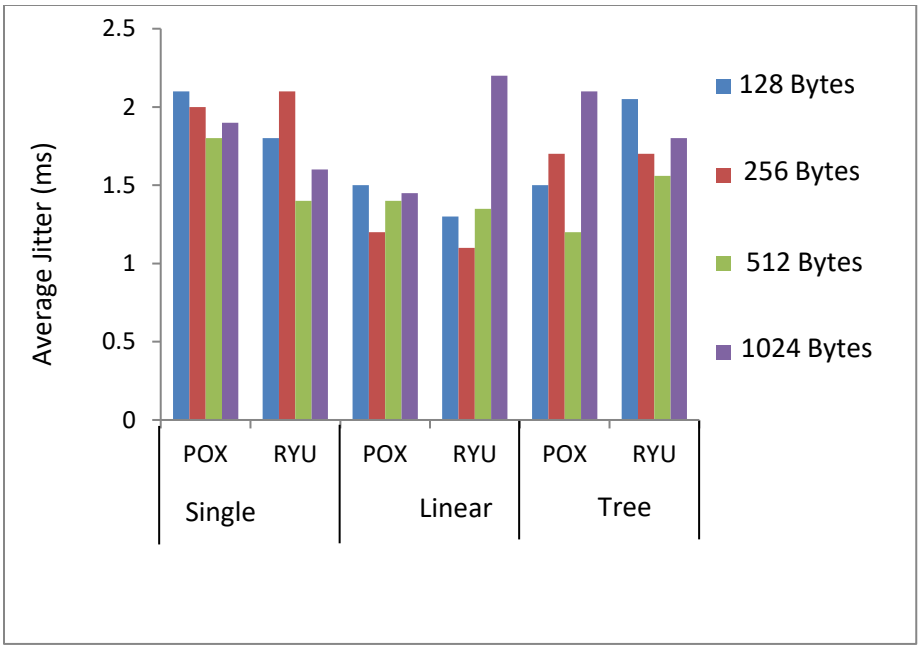

Fig. 8 Average Jitter in WL-N

C. Dropped Packet

Figure 9 illustrates how Ryu outperforms POX for the packet drop in all topologies. Additionally, trees perform improved than linear in the POX SDN controller. Figure 10 demonstrates that when the packet size is 1,026 bytes, the wickedest packet drop in the POX SDN controller with linear network topology is 13%, whereas the best packet drop is 4.5% for packet sizes of 256 bytes.

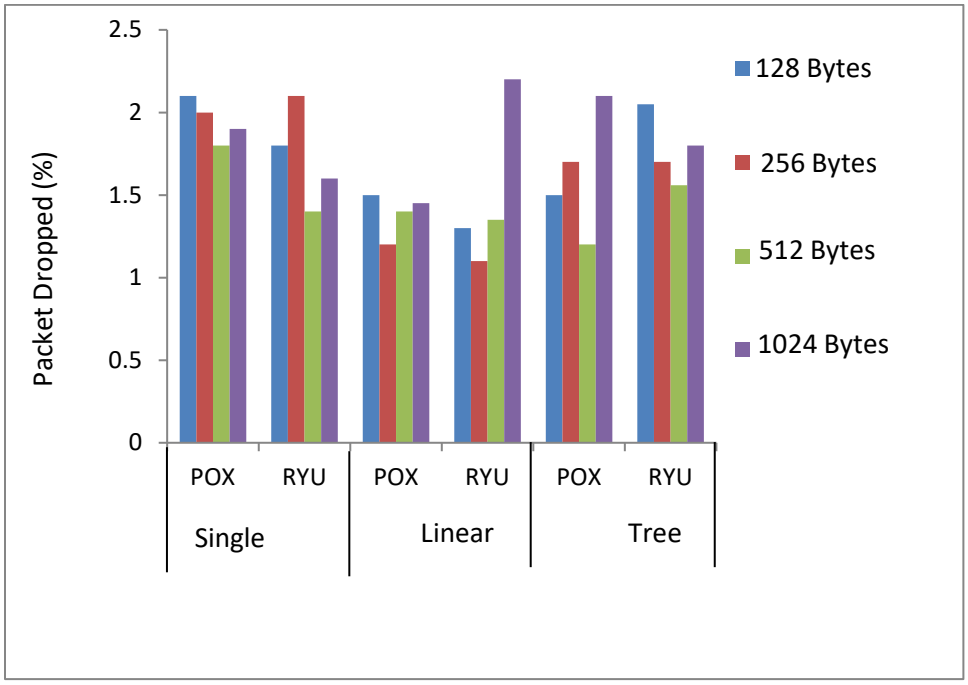

Fig. 9 Packet dropped in W-N

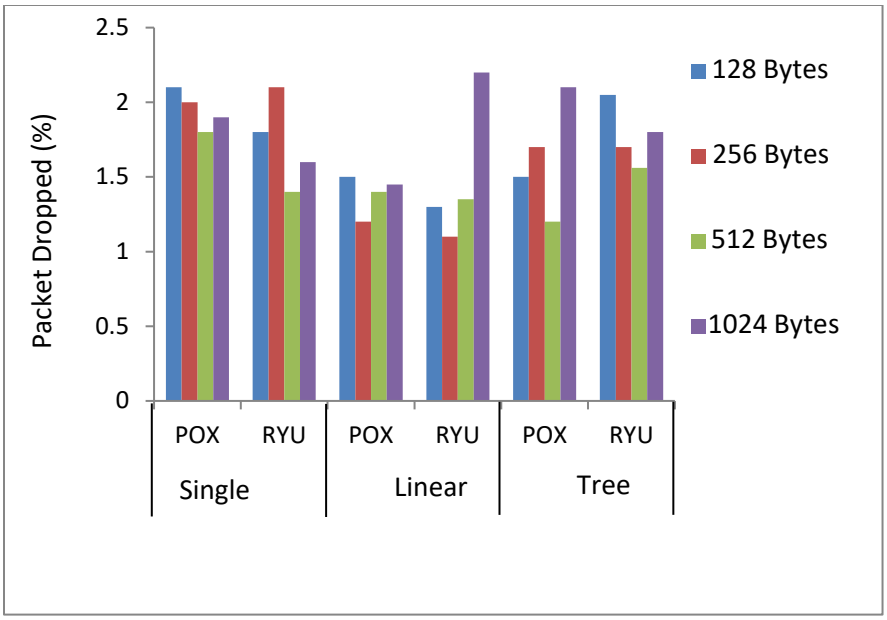

Fig. 10 Packet dropped in WL-N

D. Average Bitrate

Figure 11 demonstrates that in all situations, the average bitrate for Ryu and POX is nearly identical. Furthermore, it is shown that linear topology achieves higher bitrate than tree in POX scenarios. Figure 12 displays the 128-byte packet size is the most perverse in the wireless network. In wireless scenarios, Ryu's average bitrate is superior to POX's, and both Ryu and POX controllers give the tree topology a higher bitrate than linear.

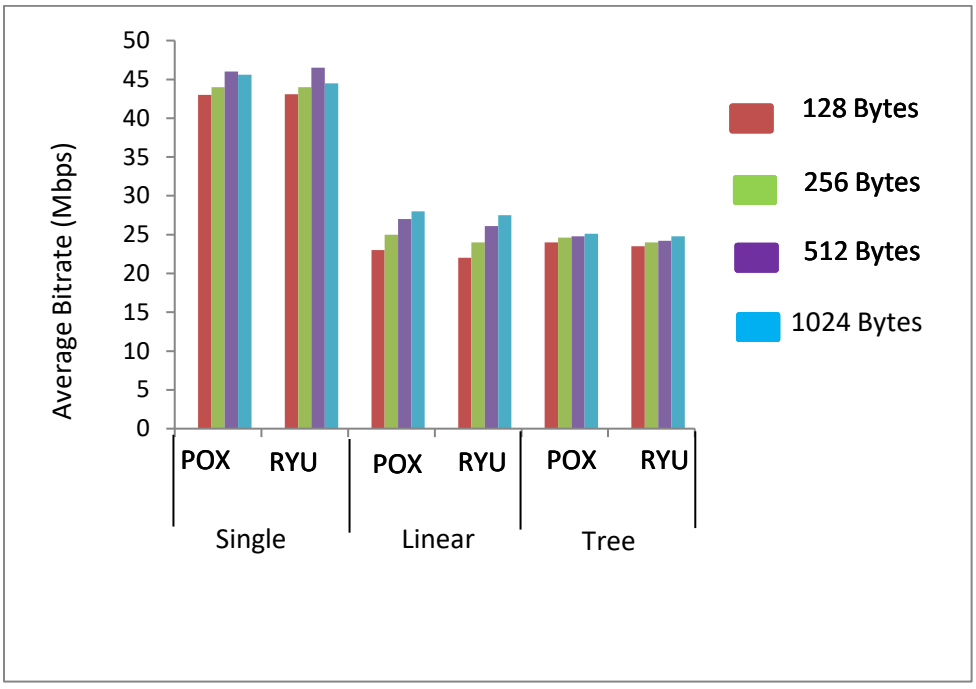

Fig. 11 Average bitrate in W-N

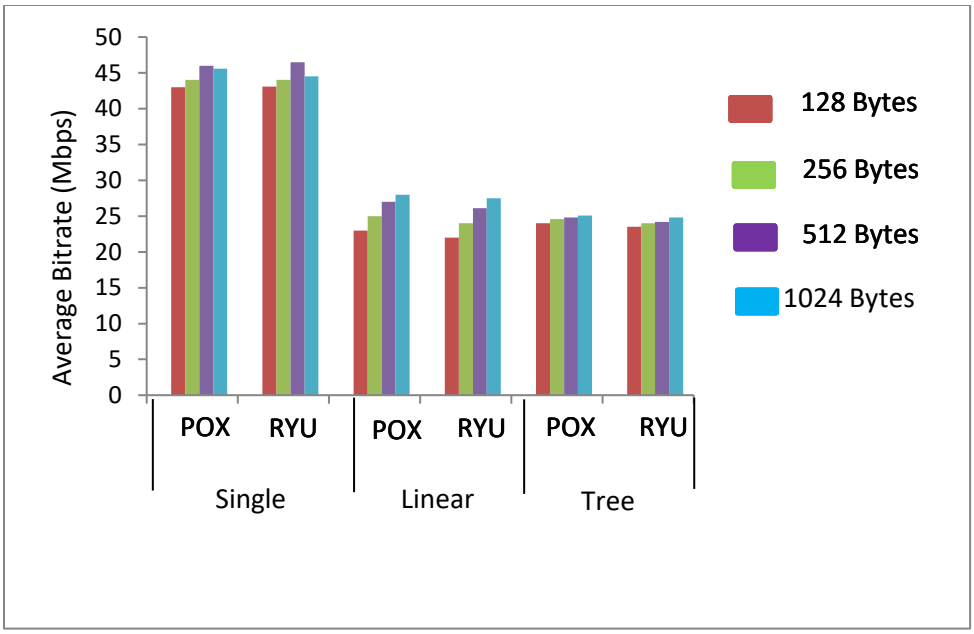

Fig. 12 Average bitrate in WL-N

VI. CONCLUSION

There has been significant technological progress in recent years, with one of the most important areas being the application of SDN in traffic analysis. Traffic analysis stands out as a key aspect within SDN's application sector. As the network's core function, the SDN controller analyzes and tracks authentic data traffic. Monitoring and analyzing real-time data traffic are crucial in any networking strategy to track the movement of data packets between users. This study involves several comparisons. Firstly, the performance of the two most well-known SDN controllers, Ryu and POX, is evaluated. Secondly, various topologies, including tree and linear, were implemented for comparison. Lastly, several packet sizes were contrasted, providing researchers with insights into the ideal packet sizes for specific applications. An objective experimental investigation based on active measurement was conducted using the Mininet-Wi-Fi emulator and the D-ITG tool. Metrics such as bitrate, packet loss, throughput, jitter, and delay were used to assess performance. According to the evaluations, Ryu consistently performed the best across all scenarios and measures. While POX's performance was relatively poor, the data collected indicated that Ryu tends to experience less jitter and delay in wireless networks as the number of packets decreases, unlike the POX controller. The Ryu SDN controller experienced a 0% packet drop while measuring packet loss with 512-byte packet sizes in both tree and linear topologies. In contrast, the POX SDN controller experienced approximately 32% packet loss with 128-byte packets in wireless network topology, marking the worst-case scenario among all configurations. Future testing may involve assessing the controller's security and robustness, particularly in sophisticated networks with multipath connections.

REFERENCES

[1] Aldabbas, H., & Amin, R. (2021). A novel mechanism to handle address spoofing attacks in SDN based IoT. *Cluster Computing, 24*(4), 3011-3026.

[2] Ali, M., Jehangiri, A. I., Alramli, O. I., Ahmad, Z., Ghoniem, R. M., Ala'anzy, M. A., & Saleem, R. (2023). Performance and Scalability Analysis of SDN-Based Large-Scale Wi-Fi Networks. *Applied Sciences, 13*(7), 4170.

[3] Askar, S., & Keti, F. (2021). Performance Evaluation of different SDN controllers: A Review.

[4] Balarezo, J. F., Wang, S., Chavez, K. G., Al-Hourani, A., & Kandeepan, S. (2022). A survey on DoS/DDoS attacks mathematical modelling for traditional, SDN and virtual networks. *Engineering Science and Technology, an International Journal, 31*, 101065.

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

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

[7] Cherian, M., & Verma, S. (2021). Integration of IoT and SDN to mitigate DDoS with RYU controller. In *Computer Networks, Big Data and IoT: Proceedings of ICCBI 202,* 673-684. Springer Singapore.

[8] Febrianto, A., & Saputra, N. (2021). Pelatihan media pembelajaran inovatif dengan videoscribe bagi guru SDN Malangrejo. *Community Empowerment, 6*(1), 24-28.

[9] Islam, M. T., Islam, N., & Refat, M. A. (2020). Node to node performance evaluation through RYU SDN controller. *Wireless Personal Communications, 112*, 555-570.

[10] Kazi, N. M., Suralkar, S. R., & Bhadade, U. S. (2021). Evaluating the performance of pox and ryu sdn controllers using mininet. In *Data Science and Computational Intelligence: Sixteenth International Conference on Information Processing, ICInPro 2021, Bengaluru, India, October 22–24, 2021, Proceedings 16* (pp. 181-191). Springer International Publishing.

[11] Keerthana, B., Balachandra, M., Hebbar, H., & Muniyal, B. (2022). Performance Comparison of Various Controllers in Different SDN Topologies. In *Expert Clouds and Applications: Proceedings of ICOECA 2021* (pp. 297-309). Springer Singapore.

[12] Kelian, V. H., Warip, M. N. M., Ahmad, R. B., Ehkan, P., Zakaria, F. F., & Ilyas, M. Z. (2023). Toward Adaptive and Scalable Topology in Distributed SDN Controller. *Journal of Advanced Research in Applied Sciences and Engineering Technology, 30*(1), 115-131.

[13] Khairi, M. H. H., Ariffin, S. H. S., Latiff, N. M. A. A., Yusof, K. M., Hassan, M. K., Al-Dhief, F. T., ... & Hamzah, M. (2021). Detection and classification of conflict flows in SDN using machine learning algorithms. *IEEE Access, 9*, 76024-76037.

[14] Khorsandroo, S., Sánchez, A. G., Tosun, A. S., Arco, J. M., & Doriguzzi-Corin, R. (2021). Hybrid SDN evolution: A comprehensive survey of the state-of-the-art. *Computer Networks, 192*, 107981.

[15] Koulouras, I., Margariti, S. V., Bobotsaris, I., Stergiou, E., & Stylios, C. (2022, November). On the Performance of SDN Controllers in Real World Topologies. In *2022 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)* (pp. 143-148). IEEE.

[16] Latif, S. A., Wen, F. B. X., Iwendi, C., Li-Li, F. W., Mohsin, S. M., Han, Z., & Band, S. S. (2022). AI-empowered, blockchain and SDN integrated security architecture for IoT network of cyber physical systems. *Computer Communications, 181*, 274-283.

[17] Ma, J., Jin, R., Dong, L., Zhu, G., & Jiang, X. (2022, May). Implementation of SDN traffic monitoring based on Ryu controller. In *International Symposium on Computer Applications and Information Systems (ISCAIS 2022)* (Vol. 12250, pp. 203-212). SPIE.

[18] Maaloul, R., Taktak, R., Chaari, L., & Cousin, B. (2018). Energy-aware routing in carrier-grade Ethernet using SDN approach. *IEEE Transactions on Green Communications and Networking, 2*(3), 844-858.

[19] Mamushiane, L., & Shozi, T. (2021, May). A QoS-based evaluation of SDN controllers: ONOS and OpenDayLight. In *2021 IST-Africa Conference (IST-Africa)* (pp. 1-10). IEEE.

[20] Mishra, A., Gupta, N., & Gupta, B. B. (2021). Defense mechanisms against DDoS attack based on entropy in SDN-cloud using POX controller. *Telecommunication systems, 77*, 47-62.

[21] Mohammadi, R., Nazari, A., Nassiri, M., & Conti, M. (2021). An SDN-based framework for QoS routing in internet of underwater things. *Telecommunication Systems, 78*(2), 253-266.

[22] Mohammed, G. A., & Aldabbagh, O. A. I. (2023). A Comparative Evaluation of the Performance of SDN Controllers (ONOS) using DOCKER Container.

[23] Nóvoa, L., Tavares, V., Nahum, C., Lins, S., & Klautau, A. (2021, July). Middleware implementation for RYU SDN Controller to manage switches in a C-RAN scenario. In *Anais do XLVIII Seminário Integrado de Software e Hardware* (pp. 19-29). SBC.

[24] Nóvoa, L., Tavares, V., Nahum, C., Lins, S., & Klautau, A. (2021, July). Middleware implementation for RYU SDN Controller to manage switches in a C-RAN scenario. In *Anais do XLVIII Seminário Integrado de Software e Hardware*, 19-29, SBC.

[25] Numan, P. E., Yusof, K. M., Marsono, M. N. B., Yusof, S. K. S., Fauzi, M. H. B. M., Nathaniel, S., & Baharudin, M. A. B. (2019). On the latency and jitter evaluation of software defined networks. *Bulletin of Electrical Engineering and Informatics, 8*(4), 1507-1516.

[26] Prabakaran, D., Nizar, S. M., & Kumar, K. S. (2021). Software-defined network (SDN) architecture and security considerations for 5G communications. In *Design methodologies and tools for 5G network development and application* (pp. 28-43). IGI global.

[27] Ramdhani, M. D., Sugiarto, B., & Rukmana, A. (2021). Simulasi Jaringan SDN menggunakan controller RYU Pada Mininet Dengan 5 Topologi Jaringan. *Jurnal FUSE-Teknik Elektro, 1*(2), 101-110.

[28] Salman, M. I. (2022). A Hybrid SDN-Multipath transmission for a Reliable Video Surveillance System. *Association of Arab Universities Journal of Engineering Sciences, 29*(2), 46-54.

[29] Saputra, Y. (2021). Analisis Performansi Software Defined Network (SDN) Controller Floodlight, Pox, Ryu, Dan Odl Pada Topologi Jaringan Universitas Islam Riau (Doctoral dissertation, Universitas Islam Riau).

[30] 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.

[31] Tivig, P. T., Borcoci, E., & Brumaru, A. (2021, October). Layer 3 Forwarder Application-Implementation Experiments Based on Ryu SDN Controller. In *2021 International Symposium on Networks, Computers and Communications (ISNCC)* (pp. 1-6). IEEE.

[32] Tseng, Y., Naït-Abdesselam, F., & Khokhar, A. (2018). A comprehensive 3-dimensional security analysis of a controller in software-defined networking. *Security and Privacy, 1*(2), e21.

[33] Umar, R., Riadi, I., & Kusuma, R. S. (2021). Mitigating sodinokibi ransomware attack on cloud network using software-defined networking (SDN). *International Journal of Safety and Security Engineering, 11*(3), 239-246.