Journal of Advanced Research Design

Journal homepage: https://akademiabaru.com/submit/index.php/ard
ISSN: 2289-7984

Performance Evaluation of Ryu, OpenDayLight and Floodlight Controllers in Diverse Software-Defined Networking Topologies

Diong Hui[1], Wei Siang Hoh[1,*], Bi Lynn Ong[2], XinPing Zhu[3], Si-Kee Yoon[1]

1 Faculty of Computing, Universiti Malaysia Pahang Al-Sultan Abdullah, 26600 Pekan, Pahang, Malaysia
2 Faculty of Electronic Engineering Technology, Universiti Malaysia Perlis, 02600 Ulu Pauh, Perlis, Malaysia
3 Faculty of Software, Harbin Institute of Information Technology, Heilongjiang 150001, China

ARTICLE INFO

Article history:
Received 6 January 2025
Received in revised form 7 February 2025
Accepted 2 May 2025
Available online 23 May 2025

Keywords:
SDN; network emulation; Mininet; Ryu; OpenDayLight; Floodlight

ABSTRACT

Software-Defined Networking (SDN) has been introduced as a new approach to networking for designing and managing computer networks. SDN architecture decouples control and data planes, enabling programmable network management. The significant component is the Controller, which is responsible for managing and distributing information to all network devices. While prior studies evaluate SDN controllers, comprehensive comparisons across multiple topologies and performance metrics remain limited. This paper addresses this gap by analysing and comparing the performance of the Ryu, OpenDayLight and Floodlight controllers under single, linear and tree network topologies using Mininet. The performance parameters considered include round-trip time (RTT), throughput, jitter and packet delivery ratio. The results indicate that in linear and tree topologies, Ryu exhibited the lowest RTT, highest bandwidth and lowest jitter compared to the ODL and Floodlight controllers. Conversely, in the single topology, Floodlight demonstrated the lowest jitter, while Ryu maintained the lowest RTT and highest bandwidth. In conclusion, Ryu is often preferred in scenarios requiring lower latency and higher throughput, particularly in more complex network topologies. However, Floodlight may still be advantageous in simpler topologies where minimizing jitter is critical. This comprehensive evaluation not only aids in controller selection but also informs future enhancements in SDN architecture and performance benchmarking methodologies, which will be implemented in modern data centres.

1. Introduction
1.1 Background

In today's world of technology, the diversified increase of new technologies such as big data, cloud computing and Internet of Things reveal that the business volume is also increasing. There has been a growing challenge over the existing network architecture with the additional data forwarding and routing devices, for example, switches and routers, which are highly outfitted with the policies and control requirements. This challenge stems from the research conducted by Liu *et al.,* [1] that

[*] *Corresponding author*
E-mail address: weisiang@umpsa.edu.my

https://doi.org/10.37934/ard.132.1.103114

the traditional network architecture is rigid yet complex to configure and manage. The control plane and data plane of traditional network forwarding devices are tightly coupled and the manual configuration of them is time consuming and error prone. The challenge is upgraded when the network is built from different vendors. In the wake of the hurdles, the idea of programmable networks was introduced by Campbell *et al.,* [2], attempted to resolve limitations of conceptual network framework. The proposed solution has brought to a series of research and eventually, the concept of Software-Defined Networking (SDN) was proposed in 2009 [3].

Several major modifications of network infrastructures in SDN to adapt the current demand [4]. The control and data planes are decoupled and communicated via southbound interface, transforming the rigid traditional network architecture into vendor-neutral, software-based network infrastructures with flexibility in self-development and evolution. This flexibility is further enhanced by the introduction of application plane which provides the programmability towards the centralized controller at the control plane, supported by northbound interface. The programmable controller acts the core of the SDN as it controls the network behaviour based on the high-level rules set. Hence, the controller used is crucial for a network performance.

1.2 Problem Statement

The rapid development of Software-Defined Networking (SDN) has led to the proliferation of diverse controllers, with Ryu, OpenDayLight (ODL) and Floodlight emerging as widely adopted solutions [5]. However, existing research exhibits critical gaps:

i. limited comparative analysis of these controllers' inherent features (e.g., architectural design, protocol support, scalability)
ii. insufficient exploration of their performance across varied network topologies (e.g., single, linear, tree)
iii. fragmented evaluation of critical metrics such as latency, throughput, jitter and packet delivery ratio under topology-specific conditions.

Prior studies often focus on isolated parameters or simplified topologies, neglecting the interplay between controller architecture, topology complexity and multi-metric performance. This work addresses these gaps by conducting a systematic evaluation of Ryu, ODL and Floodlight across single, linear and tree topologies, providing empirical insights to guide controller selection based on network requirements and operational environments.

1.3 Contributions

The key contributions of this study:

i. <u>Comprehensive Evaluation:</u> Ryu, ODL and Floodlight are analysed under diverse topologies, incorporating both TCP and UDP traffic to assess throughput, latency, jitter and packet delivery ratio.
ii. <u>Architectural Insights:</u> Performance outcomes are correlated with controller design principles, demonstrating Ryu's Python-based asynchronous processing excels in complex topologies, whereas Floodlight's Java-based modularity benefits simpler setups.

iii. Practical Guidelines: Evidence-based recommendations for controller selection are provided, tailored to network scale and application requirements, such as prioritizing Ryu for data centres and Floodlight for real-time communication.

2. Literature Review

The performance of SDN controllers has been extensively studied, yet existing literature exhibits critical limitations. Early work by Mamushiane *et al.,* [5] has evaluated the performance of Ryu, Floodlight, ONOS and OpenDaylight in terms of throughput and latency using the same topology. The experiment is setup by varying the number of switches in first scenarios and the number of MACs in another. In this paper, ONOS is found to have the best performance in terms of throughput and Ryu is best in terms of least latency.

The performance analysis is done by Bholebawa *et al.,* [6] using Floodlight, a Java-based controller and POX, a python-based controller. The RTT and throughput of the controllers are analysed under single, linear, tree and custom topologies. This research found that Floodlight outperforms POX in both metrics.

Work done by Rowshanrad *et al.,* [7] is to evaluate and compare the performances of the SDN controllers namely Floodlight and OpenDayLight in different topologies which are single, linear and tree topologies. The performance metrics involved are traffic loads, delay and packet loss. The results demonstrate that, in terms of latency, OpenDaylight performs better than Floodlight in both low-loaded networks and tree topology of medium-loaded networks. In heavily-loaded networks, Floodlight can surpass OpenDaylight in terms of latency and packet loss for linear topology and tree topology, respectively.

In Gupta *et al.,* [8] the effects of several SDN controllers on SDN are analysed as controller is the core element in SDN that influences the network performance. The work done by researches in past 9 years is used in the study. The functionalities and features of several controllers such as Beacon, Faucet, Nox, POX, ONOS, OpenDayLight, Ryu, Floodlight, Trema and MUL are discussed. This paper concluded that different controllers have their suitable use cases and the selection of the controller is reflected to its intention.

From Askar *et al.,* [9], the performance of POX and Ryu controllers are evaluated in the aspect of throughput, round-trip time and jitter using single topologies only. It was found that Ryu controller performed better than POX controller.

The work conducted by Chouhan *et al.,* [10] is to compare the performance between Ryu and Floodlight controller under several network topologies such as single, linear, tree, torus and custom. The metrics involved are latency, throughput, packet loss and jitter. It was found that the performance of Ryu is better than Floodlight in all topologies of all metrics except the latency and jitter of torus topology and packet loss of linear topology.

Arahunashi *et al.,* [11] has compared the performance between Ryu, OpenDayLight and ONOS controllers using linear, tree and mesh topologies. The parameters considered are average latency and throughput. Through the work done, Floodlight controller is found to have best performance.

In summary, the network performance of several controllers is analysed in different network parameters using different network topologies. The network performance SDN controllers are selected according to their popularity and previous work performances. In this paper, the popular parameters according to previous work which are round-trip time, throughput, jitter and packet delivery ratio will be all analysed under various network topologies which are single, linear and tree topologies for a more comprehensive review. This approach not only validates prior findings but also reveals new insights, such as Ryu's superiority in tree topologies and Floodlight's efficiency in minimizing jitter.

3. Methodology

3.1 Experiment Setup

The Mininet and the controllers are installed on the laptop configured with AMD Ryzen 7 4800H CPU @2.90 GHz processor, 512SSD and 16GB RAM running the Linux Operating System Ubuntu version 20.04.6 LTS-64 bit.

3.2 Features of Ryu, OpenDayLight and Floodlight

Table 1 provides the overview of the features supported by Ryu, OpenDayLight and Floodlight controllers. This table is updated from several works being compared to verity the features of the controllers [12-15].

Table 1
Feature comparison

Feature	Ryu	OpenDayLight	Floodlight
Southbound Interfaces	OpenFlow 1.0, 1.2, 1.3, 1.4, 1.5, NETCONF, OFCONFIG, OVSDB	OpenFlow 1.0, 1.3, 1.4, 1.5, NETCONF, OFCONFIG, YANG Model, SNMP, OVSDB, PCEP, LISP, BGP/LS	OpenFlow 1.0, 1.1, 1.2, 1.3, 1.4, 1.5
REST API	Yes (For SB only)	Yes	Yes
GUI	Yes (Initial Phase)	Web-based	Java/Web-based
Modularity	Medium	Medium	High
Productivity	Medium	Medium	Medium
OS Support	Most supported-on Linux	Linux, MAC, Windows	Linux, MAC, Windows
Partnership	Nippo Telegraph and Telephone Corporation (NTT)	Linux Foundation with membership covering more than 40 companies	Big Switch Networks
Documentation	Medium	Very good	Good
Programming Language	Python	Java	Java
Distributed/Centralized	Centralized	Distributed	Centralized
Virtualization	Open vSwitch and Mininet	Open vSwitch and Mininet	Open vSwitch and Mininet
Multi- threading support	Yes	Yes	Yes
Application domain	Campus	Transport-SDN WAN, data centre	Campus
SDN deployment scale	Small/Medium		Small
Cloud orchestrator support	Yes	Yes	No

3.3 Mininet

Mininet is an open-source, lightweight network virtualization emulator that is crucial for SDN research, particularly for creating the SDN network topologies. Mininet allows the emulation of switches, hosts and links to stimulate the network environments via Linux containers. Without requiring the physical network infrastructures, Mininet provides a versatile environment for network building, testing and education purposes such as experimenting with different network configurations and protocols.

Mininet supports a variety of features to stimulate the network environments. It supports Python scripts, allowing the customization of the functionalities. Users can use default or develop custom network topologies in Mininet, saving time in building topology. Mininet supports OpenFlow protocol for the communication between control plane and data plane. With the OpenFlow protocol, the network built can be controlled in aspects such as traffic forwarding, topology discovery and Quality of Service. In addition, Mininet provides tools which aid in monitoring the network traffic and performance analysis.

3.4 Network Topologies

Network topology describes the structure of the network physically and logically. Mininet is an emulation tool with various default topologies [16]. The performances of Ryu, OpenDayLight and Floodlight controllers are tested using the default single, linear and tree topologies in Mininet.

Single topology is built with 1 switch and 27 hosts and Figure 1 shows the single topology generated.

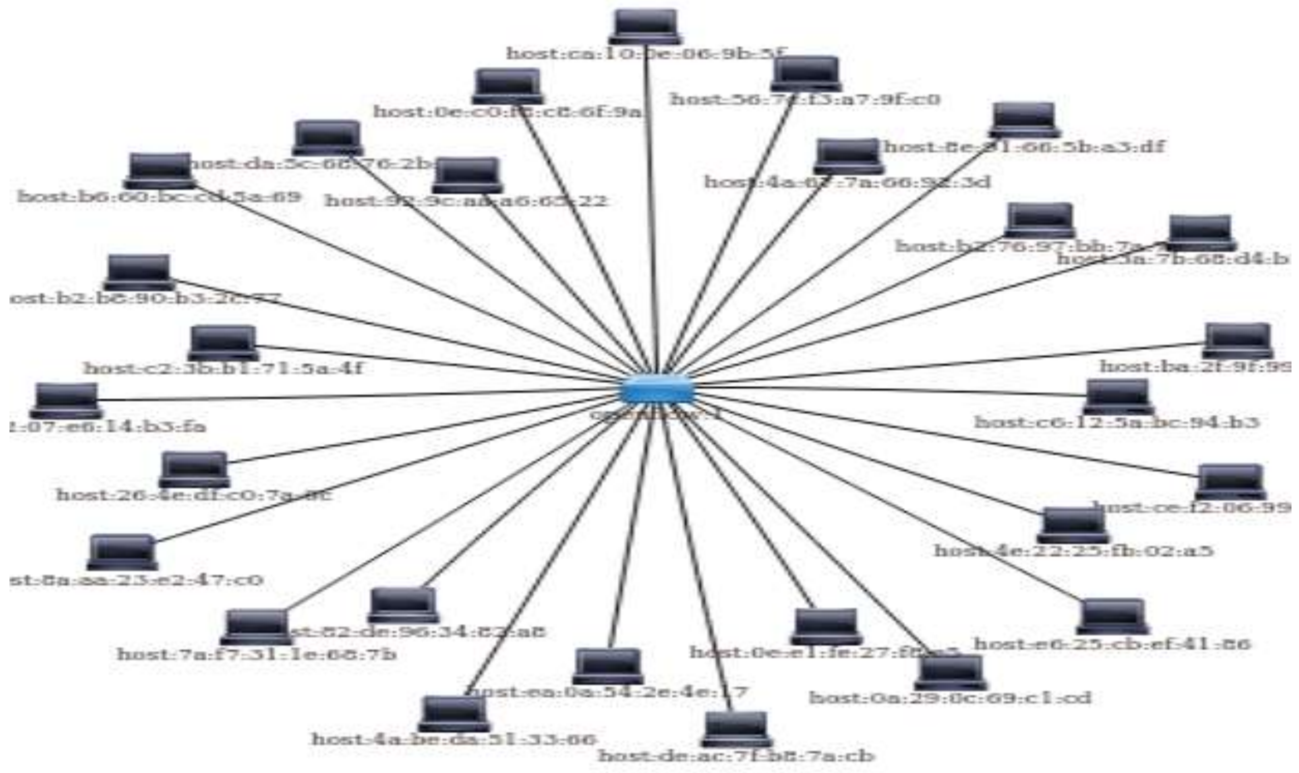


Fig. 1. Single topology view in OpenDayLight

Linear topology is built with 9 switches and 27 hosts and Figure 2 shows the linear topology generated.

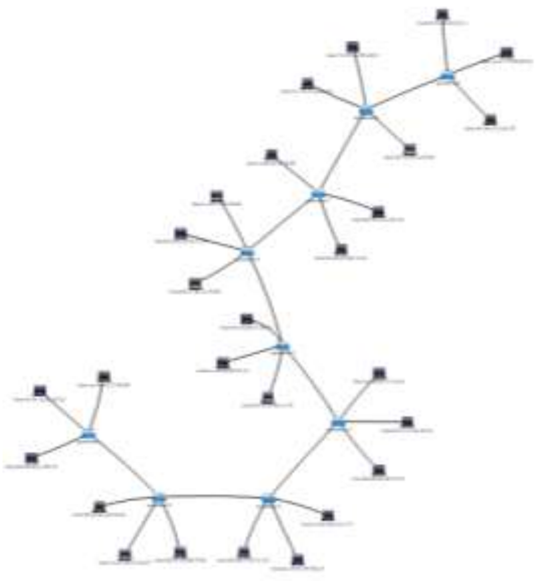
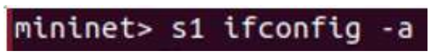


Fig. 2. Linear topology view in OpenDayLight

Tree topology is built with the depth (number of levels of the switch) of 3 and the fanout (the number of output ports) of 3. Hence, 13 switches and 27 hosts are generated. Switch 1 of the tree topology is set as root node as shown in Figure 3.



```
mininet> s1 ifconfig -a
```

Fig. 3. Root node command

Figure 4 shows the tree topology generated.

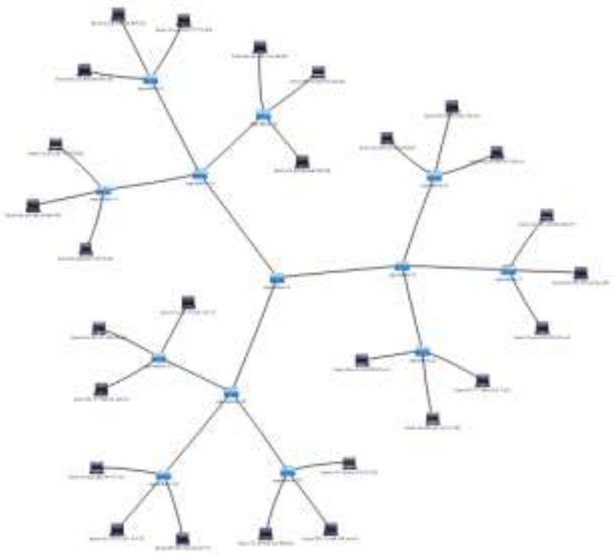


Fig. 4. Tree topology view in OpenDayLight

3.5 Performance Parameters
3.5.1 Latency

The latency is measured using the round-trip time (RTT) which is the time taken for the data packet to reach the destination and return to the source. As shown by Shirvar *et al.,* [17] the round-trip time can be calculated using the formula RTT = Time of packet received – Time of packet departed.

To collect the average RTT, the latency tests of Ryu, OpenDayLight and Floodlight controllers are conducted by executing the ICMP connectivity. Average RTT of each controller under each topology will be collected when 10 packets are ping from host 1 to host 27.

3.5.2 Throughput

From the Punithavathani *et al.,* [18], the throughput can be calculated using the formula *T=P/t* where *T* represents throughput, *P* represents packet size of the data transferred and *t* represents time cost in transferring.

To evaluate this parameter, Iperf3 command is executed in Mininet. The average TCP and UDP bandwidth of each controller under each topology are collected within 10 intervals. In bandwidth test, host 1 will set as the client and host 27 as the server. The packets will be sent as many from the client to server.

3.5.3 Jitter

Jitter of each controller under each topology can be observed when the UDP bandwidth is executed. Calculated as the variance in packet arrival times during UDP tests.

3.5.4 Packet delivery ratio

As shown in Fazeldehkordi *et al.,* [19], the packet delivery ratio can be calculated using the formula PDR=R/S where PDR represents packet delivery ratio, R represents total number of packets received and S represents total number of packets send. The total packet loss and total packet sent of each controller under each topology can be collected when the UDP bandwidth is executed.

4. Results

The RTT, TCP throughput, UDP throughput, jitter and packet delivery ratio are tested using Mininet connected to Ryu, OpenDayLight and Floodlight controllers sequentially under single, linear and tree topologies. The hosts involved are host 1 and host 27.

From the graph shown in Figure 5, when the RTT is compared from the aspect of controllers, Ryu controller has the lowest average RTT, followed by OpenDayLight controller and then Floodlight controller in single and linear topologies. In tree topology, the average RTT of Ryu controller remains the lowest, followed by the Floodlight controller and then the OpenDayLight controller. In another hand, when average RTT is compared from the aspect of topologies, single topology has the lowest average RTT, followed by tree topology and then the linear topology for both Ryu and OpenDayLight controllers. However, in Floodlight controller, tree topology has a lower average RTT than single topology and linear topology remains to have the highest average RTT.

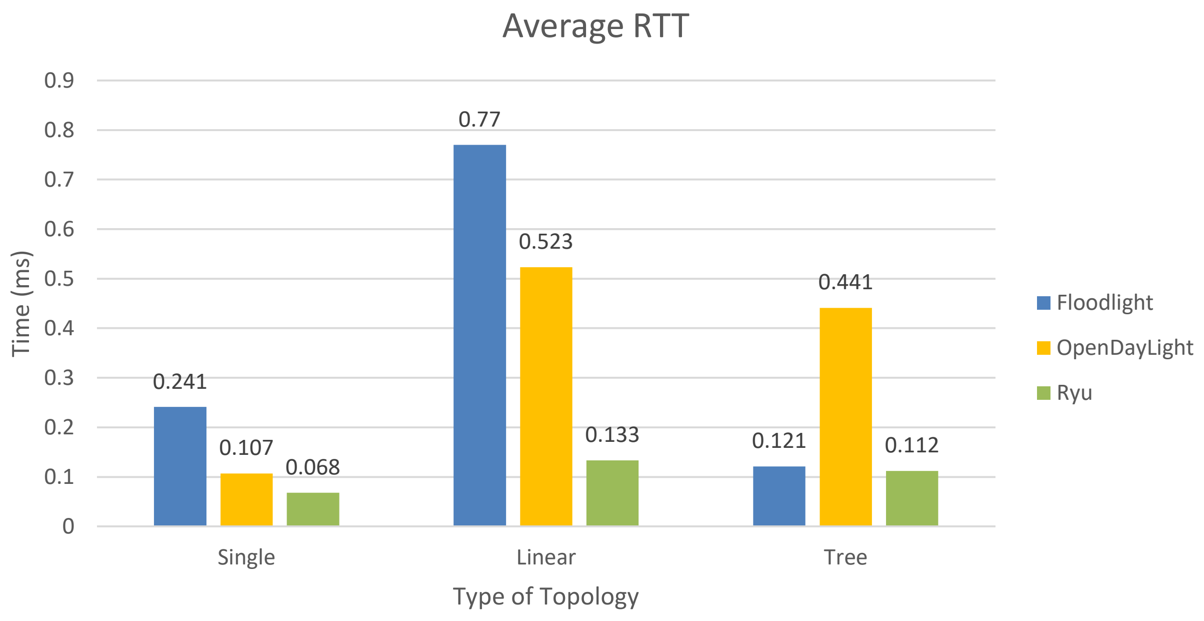


Fig. 5. RTT comparison

From the graph shown in Figure 6, when the TCP bandwidth is compared from the aspect of controllers, Ryu controller has the highest average TCP bandwidth, followed by Floodlight controller and then OpenDayLight controller in linear and tree topologies. In single topology, the average TCP bandwidth of Ryu controller is still the highest, followed by the OpenDayLight controller and then the Floodlight controller. In another hand, when average TCP bandwidth is compared from the aspect of topologies, single topology has the highest average bandwidth, followed by tree topology and then the linear topology for both OpenDayLight and Floodlight controllers. However, in Ryu controller, tree topology has a highest average bandwidth than single topology and linear topology remains to have the lowest average bandwidth.

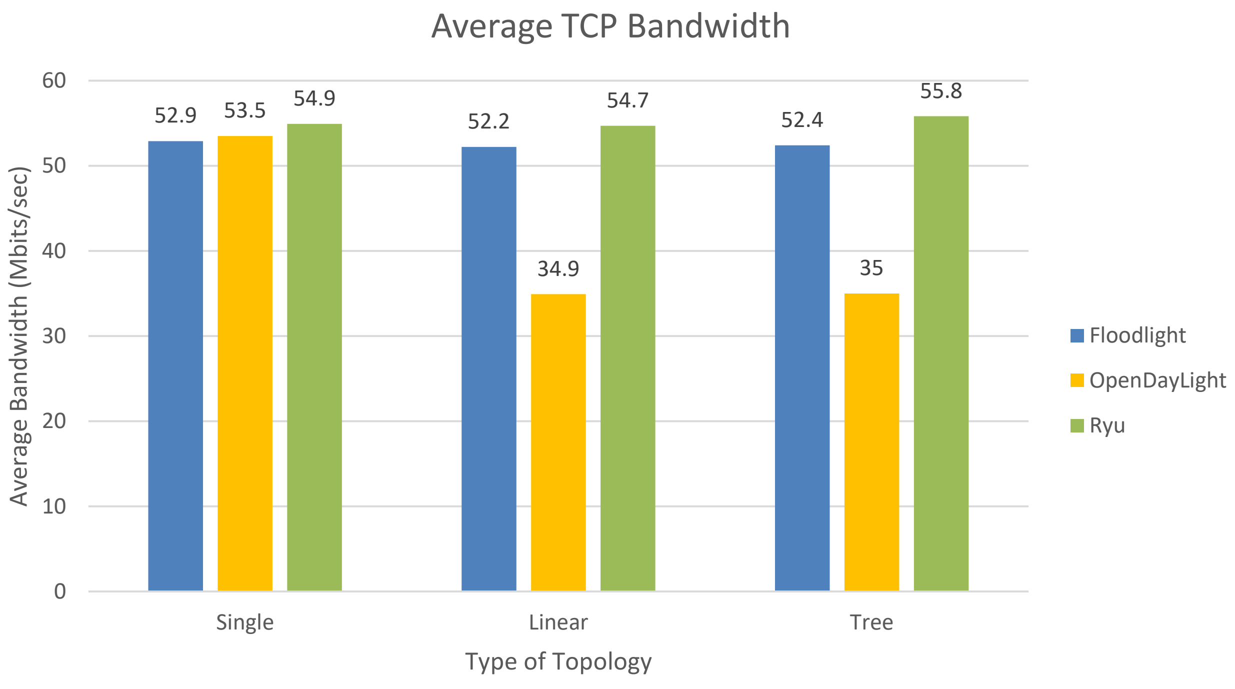


Fig. 6. TCP bandwidth comparison

From the graph shown in Figure 7, when the UDP bandwidth is compared from the aspect of controllers, Ryu controller has the highest average UDP bandwidth, followed by Floodlight controller and then OpenDayLight controller in linear and tree topologies. In single topology, the average TCP bandwidth of Ryu controller is still the highest, followed by the OpenDayLight controller and then the Floodlight controller. In another hand, when average TCP bandwidth is compared from the aspect of topologies, single topology has the highest average UDP bandwidth, followed by tree topology and then the linear topology for all controllers.

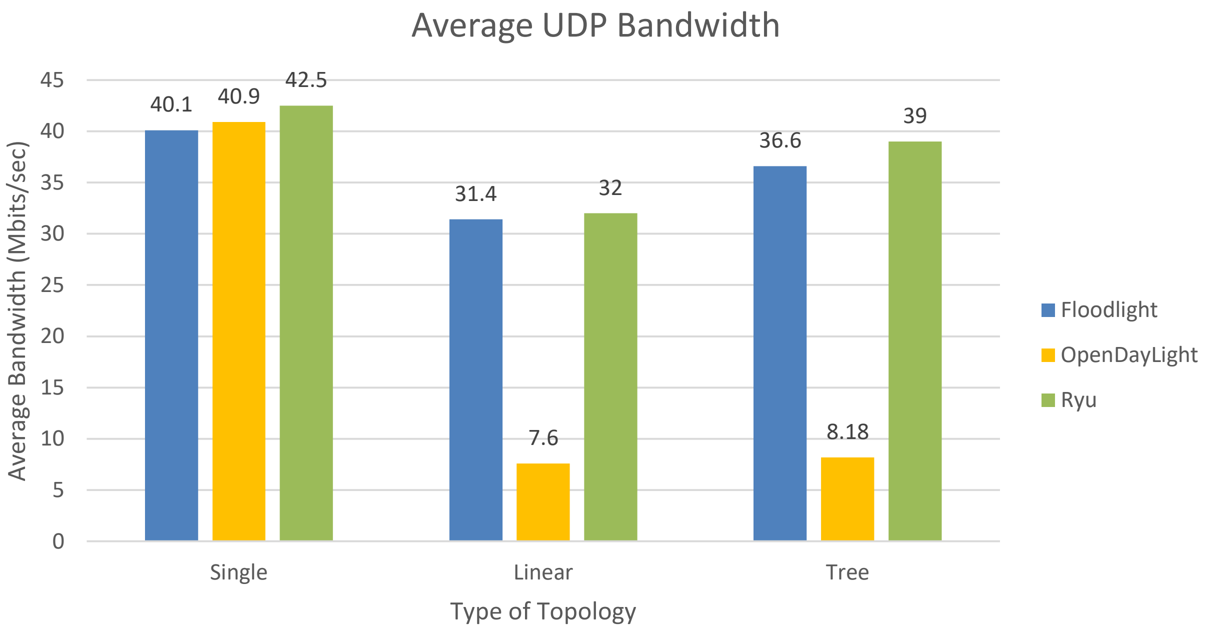


Fig. 7. UDP bandwidth comparison

Based on Figure 5 until Figure 7, Ryu exhibited superior performance in complex topologies, achieving the lowest RTT (0.068 ms in single topology), highest TCP bandwidth (55.8 Mbps in tree topology) and highest UDP bandwidth (42.5 Mbps in single topology). This is attributed to its lightweight Python architecture, which minimizes processing overhead. In contrast, Floodlight's Java-based design incurred higher latency (0.77 ms in linear topology) but delivered stable TCP throughput (52.2 Mbps in linear topology), ideal for environments prioritizing reliability over speed. ODL, despite its modularity, lagged in both metrics due to resource-intensive features like SNMP and BGP support.

From the graph shown in Figure 8, when the jitter is compared from the aspect of controllers, Ryu controller has the lowest average jitter, followed by OpenDayLight controller and then Floodlight controller in linear and tree topologies. In single topology, the average jitter of Floodlight controller is the lowest, followed by the Ryu controller which remains lower than OpenDayLight controller. In another hand, when average jitter is compared from the aspect of topologies, single topology has the lowest average jitter, followed by tree topology and then the linear topology for both the OpenDayLight and Floodlight controllers. However, in Ryu controller, tree topology has a lower average jitter than single topology and linear topology remains to have the highest average bandwidth.

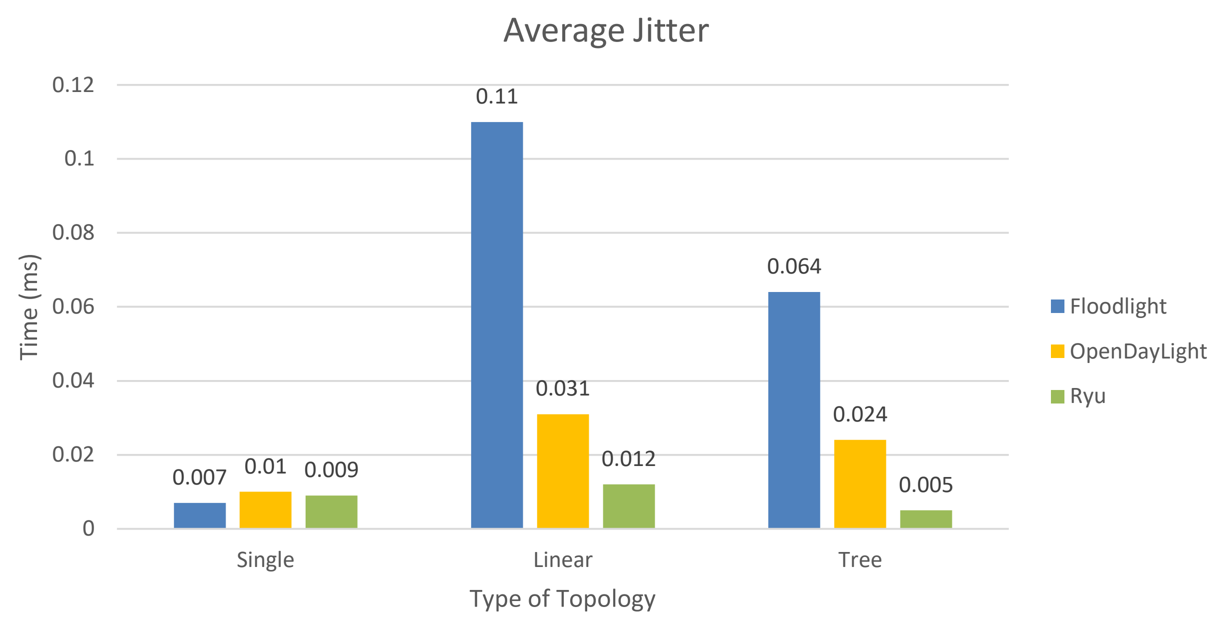


Fig. 8. Jitter comparison

From the graph shown in Figure 9, the packet delivery ratio of all controllers under all network topology is 1, which means that there is no packet loss. The result is due to the small network, running on the same machines.

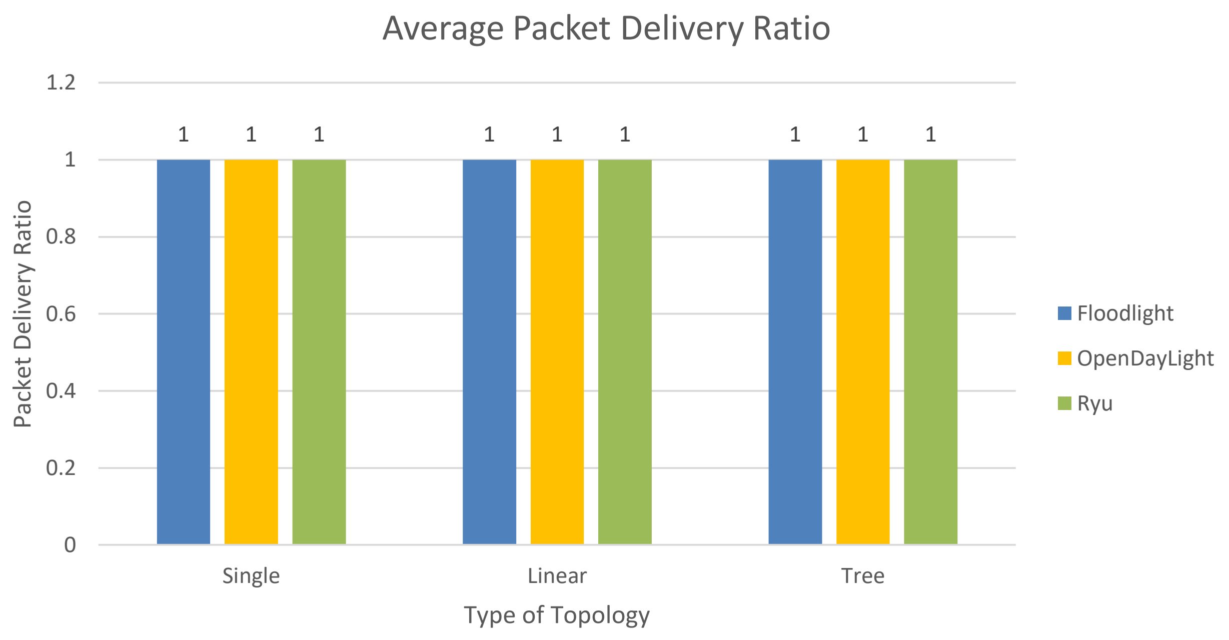


Fig. 9. Packet delivery ratio comparison

Based on Figures 8 and 9, Floodlight outperformed others in single topologies, achieving the lowest jitter (0.007 ms), a critical factor for VoIP and video streaming. However, its performance degraded in linear and tree topologies, where Ryu maintained consistent jitter levels (0.009 ms). All controllers achieved a 100% packet delivery ratio, reflecting Mininet’s controlled environment.

Single topologies favoured Floodlight due to minimal switch-controller interactions, while Ryu thrived in tree topologies by efficiently managing distributed flows. ODL's performance suffered in all topologies, highlighting the trade-offs of modular design.

5. Conclusion & Future Work

This study offers a comprehensive analysis of SDN controller performance, highlighting the critical influence of network topology and architectural design on operational efficiency. The findings demonstrate that Ryu, with its lightweight Python-based architecture, excels in scalable, high-throughput environments such as data centres, delivering superior performance in complex topologies. Conversely, Floodlight's Java-based modularity proves advantageous in simpler topologies like enterprise LANs, where minimizing jitter for latency-sensitive applications is paramount. Meanwhile, OpenDayLight's modular framework, while versatile, necessitates optimization to mitigate resource overhead. Future research should prioritize extending these evaluations to real-world SDN deployments with heterogeneous traffic patterns to validate scalability under practical conditions. Additionally, exploring machine learning-driven dynamic controller adaptation could enhance responsiveness in fluctuating network environments. Further investigation into hybrid topologies, such as fat-tree architectures, would provide deeper insights into scalability limits and controller robustness in large-scale, distributed networks. These directions promise to refine SDN benchmarking methodologies and inform the development of adaptive, topology-aware controller solutions.

References

[1] Liu, Fanglin, Godfrey Kibalya, S. V. N. Santhosh Kumar and Peiying Zhang. "Challenges of traditional networks and development of programmable networks." In *Software defined internet of everything*, pp. 37-61. Cham: Springer International Publishing, 2021. https://doi.org/10.1007/978-3-030-89328-6_3

[2] Campbell andrew T., Herman G. De Meer, Michael E. Kounavis, Kazuho Miki, John B. Vicente and Daniel Villela. "A survey of programmable networks." *ACM SIGCOMM Computer Communication Review* 29, no. 2 (1999): 7-23. https://doi.org/10.1145/505733.505735

[3] Zhang, Zhao, Hailong Li, Siqi Dong and Lei Hu. "Software defined networking (SDN) research review." In *2018 International Conference on Mechanical, Electronic, Control and Automation Engineering (MECAE 2018)*, pp. 291-300. Atlantis Press, 2018. https://doi.org/10.2991/mecae-18.2018.129

[4] Waseem, Quadri, Wan Isni Sofiah Wan Din, Afrig Aminuddin, Muzammil Hussain Mohammed and Rifda Faticha Alfa Aziza. "Software-defined networking (SDN): a review." In *2022 5th international conference on information and communications technology (ICOIACT)*, pp. 30-35. IEEE, 2022. https://doi.org/10.1109/ICOIACT55506.2022.9972067

[5] Mamushiane, Lusani, Albert Lysko and Sabelo Dlamini. "A comparative evaluation of the performance of popular SDN controllers." In *2018 Wireless Days (WD)*, pp. 54-59. IEEE, 2018. https://doi.org/10.1109/WD.2018.8361694

[6] Bholebawa, Idris Z. and Upena D. Dalal. "Performance analysis of SDN/OpenFlow controllers: POX versus floodlight." *Wireless Personal Communications* 98 (2018): 1679-1699. https://doi.org/10.1007/s11277-017-4939-z

[7] Rowshanrad, Shiva, Vajihe Abdi and Manijeh Keshtgari. "Performance evaluation of SDN controllers: Floodlight and OpenDaylight." *IIUM Engineering Journal* 17, no. 2 (2016): 47-57. https://doi.org/10.31436/iiumej.v17i2.615

[8] Gupta, Neelam, Sarvesh Tanwar, Sumit Badotra and Sunny Behal. "Performance Analysis of SDN controller." *International Journal of Performability Engineering* 18, no. 8 (2022): 537. https://doi.org/10.23940/ijpe.22.08.p1.537544

[9] Askar, Shavan and Faris Keti. "Performance evaluation of different SDN controllers." (2021): 67-80.

[10] Chouhan, Ravindra Kumar, Mithilesh Atulkar and Naresh Kumar Nagwani. "Performance comparison of Ryu and floodlight controllers in different SDN topologies." In *2019 1st International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE)*, pp. 188-191. IEEE, 2019. https://doi.org/10.1109/ICATIECE45860.2019.9063806

[11] Arahunashi, Arun K., S. Neethu and HV Ravish Aradhya. "Performance analysis of various sdn controllers in mininet emulator." In *2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, pp. 752-756. IEEE, 2019. https://doi.org/10.1109/RTEICT46194.2019.9016693

[12] Khondoker, Rahamatullah, Adel Zaalouk, Ronald Marx and Kpatcha Bayarou. "Feature-based comparison and selection of Software Defined Networking (SDN) controllers." In *2014 world congress on computer applications and information systems (WCCAIS)*, pp. 1-7. IEEE, 2014. https://doi.org/10.1109/WCCAIS.2014.6916572

[13] Salman, Ola, Imad H. Elhajj, Ayman Kayssi and Ali Chehab. "SDN controllers: A comparative study." In *2016 18th mediterranean electrotechnical conference (MELECON)*, pp. 1-6. IEEE, 2016. https://doi.org/10.1109/MELCON.2016.7495430

[14] Li, Yanzhen, Xiaobo Guo, Xue Pang, Bo Peng, Xiaoyue Li and Peiying Zhang. "Performance analysis of floodlight and ryu SDN controllers under mininet simulator." In *2020 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, pp. 85-90. IEEE, 2020. https://doi.org/10.1109/ICCCWorkshops49972.2020.9209935

[15] Chandroth, Jisi, Byeong-Hee Roh and Jehad Ali. "Performance analysis of python based SDN controllers over real internet topology." In *2022 Thirteenth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 283-288. IEEE, 2022. https://doi.org/10.1109/ICUFN55119.2022.9829591

[16] Kumar, Deepak and Manu Sood. "Software defined networks (SDN): experimentation with Mininet topologies." *Indian Journal of Science and Technology* 9, no. 32 (2016): 1-7. https://doi.org/10.17485/ijst/2016/v9i32/100195

[17] Shirvar, Arash and Bhargavi Goswami. "Performance comparison of software-defined network controllers." In *2021 International conference on advances in electrical, computing, communication and sustainable technologies (ICAECT)*, pp. 1-13. IEEE, 2021. https://doi.org/10.1109/ICAECT49130.2021.9392559

[18] Punithavathani, D. Shalini and K. Sankaranarayanan. "IPv4/IPv6 transition mechanisms." *European Journal of Scientific Research* 34, no. 1 (2009): 110-124.

[19] Fazeldehkordi, Elahe, Iraj Sadegh Amiri, Oluwatobi Ayodeji Akanbi and M. Neely. "A Study of Black Hole Attack Solutions." (2016).