EAI Endorsed Transactions
on Internet of Things
Research Article EAI.EU

Comparative Performance Analysis of SDN Controllers Across Fat -Tree and Mesh Data Center Topologies

B.V. Prasanthi[1,2*], P. Chenna Reddy[2]

[1]Vishnu Institute of Technology
[2]JNTUA College of Engineering, Anantapuramu, Andhra Pradesh, India

Abstract

Software Defined Networking (SDN) plays a major role in today's modern cloud and 5G systems. It separates control from forwarding and allows operators to manage the network through programmable controllers. The actual performance of an SDN deployment is dependent on the selected controller and underlying topology. This work examines eight widely used controllers POX, RYU, Floodlight, Faucet, Beacon, ONOS, OpenDaylight (ODL), and default OpenFlow controller under topologies like Fat-Tree and Mesh. Using Mininet, we evaluated each controller on various network instances and recorded round-trip time, latency, throughput, and bandwidth usage. Across all experiments, ONOS and ODL achieved lower delay and higher throughput than the other controllers. Controllers such as RYU and floodlight performed steadily under moderate load, whereas POX and reference controller show higher delay and low throughput as network size increases. The distributed processing and efficient handling of ONOS and ODL provide significant advantages in large Fat-Tree and Mesh environments.

Keywords: Software-Defined Networking, OpenFlow, ONOS, OpenDaylight, Fat-Tree, Mesh Topology, Data Center Networks, Mininet, Controller Performance

Received on 16 June 2026, accepted on 07 September 2026, published on 28 September 2026

doi: 10.4108/eetiot.13550

1. Introduction

The rapid use of global networks is rising because of cloud computing, video streaming, edge services and 5G-based applications. In context of large data centers, telco clouds and multisite enterprise networks, millions of servers and network devices need to be operated as a single logical system. The conventional distributed control protocols (i.e. outdated routing and switching systems) find it difficult to provide rapid reconfiguration, refined traffic routing, and application-sensitive policies at this scale. Software Defined Networking (SDN) covers these limitations, which consists of decoupling the control plane and data plane permitting the network operators to program the forwarding devices based on a logically central controller [1, 2]. The SDN controller provides applicability to programmable abstractions that expose southbound protocols like OpenFlow to switches. The performance, scalability and resilience of the controller have a direct influence on the behavior of the entire network [3]. Simultaneously, data center and core network designs have been narrowing down to several topologies that are common. The fat-tree networks, which are typically implemented as the most prevalent in the modern data center. They are characterized by a high bisection bandwidth, path diversity and predictable scaling [4]. Mesh topologies (full or partial) remain popular in metro aggregation, carrier backbones, and inter-data-center fabrics, where multiple redundant paths are used for fast restoration and traffic engineering (TE). As these

*Corresponding author. Email: prasanthibeera@gmail.com

topologies grow, ensuring that the controller can maintain acceptable RTT, latency, throughput, and bandwidth utilization under realistic traffic is crucial [5]. Several studies have evaluated SDN controllers, most of them focused on small networks, limited topologies, or isolated metrics [6]. However, the most relevant deployment scenarios for modern cloud and carrier networks involve Fat-Tree and Mesh, and the behavior of controllers in such environments needs dedicated analysis [7].

2. Related Work

SDN's central idea is to delegate network intelligence to a logically centralized controller that maintains a global view of the network and programs forwarding devices via southbound APIs. This approach simplifies network management and supports rapid innovation which enables advanced functions such as intent-based networking, dynamic QoS with refined security policies [8, 9]. Controllers can be centralized (e.g., simple academic controllers like POX) or distributed or clustered (e.g., ONOS and OpenDaylight) [10]. Distributed controllers need to share and coordinate, especially in large topologies like Fat-tree and mesh, where thousands of switches and millions of flows must be handled. We consider a k-ary Fat-Tree with parameter k chosen according to the emulated scale (e.g., k = *4, 8, 16*). The topology consists of $(k^2 / 4)$ core switches, $(k^2 / 2)$ aggregation switches, $(k^2 / 2)$ edge (ToR) switches, and $(k^3 / 4)$ hosts. Each edge switch connects to hosts and aggregation switches, and each aggregation switch connects upwards to core switches. Multiple equal-cost paths exist between any two hosts, allowing the controller to load-balance flows [11]. Mininet is implemented either custom topology scripts or built-in Fat-Tree generators [12, 13]. Network is connected in small, medium, and large variants by varying k. For Mesh networks, we consider an N×N grid of switches or a near-complete partial mesh, depending on the scenario. Hosts may be attached at the grid edges or uniformly across nodes. Important characteristics of these topologies include multiple redundant routes between any two points, high potential for alternative paths under failures and increased path diversity, which can be exploited by TE applications [14]. Mesh topologies are especially relevant for backbone and inter-DC networks where robust connectivity and failover are priorities [15]. The limitations of existing evaluations focus on simple topologies like single switch, line, ring, or basic trees. It considers only small network sizes and/or traffic patterns and evaluate a few metrics, it is difficult to understand a trade-off, and the effects of time [16]. Recent studies provide valuable background information that does not support performance-intensive applications such as hyperscale data centers and 5G core networks with Fat-Tree and Mesh topologies [17]. In our recent research study, we compared controller performance and placement solutions in star, linear and ring topologies and gained some insights into the behavior of the controller in relatively simpler topologies [18]. On the other hand, the present study focuses on Fat-Tree and 6×6 mesh topologies where path diversity and scale of the network are increased, thereby examining the performance of the controllers. Evaluations of these topologies at larger scales are still limited, and this study helps to address that gap. Experimental environments and methodology are discussed in the next section.

3. Methodology

The proposed system architecture is organized into four coordinated layers. Each layer is responsible for a specific stage of the SDN controller evaluation process as in Figure 1. At the top, the application and orchestration layer deals with setting up an experiment, traffic generation, and performance visualization. It ensures that the test situations are same in various controllers and topologies. The next layer is control plane layer, which contains the SDN controllers like POX, RYU, Floodlight, Faucet, Beacon, ONOS, ODL and default OpenFlow controller. These are used as selection module switches on the corresponding experiment controller. The data plane layer and emulation layers use Mininet to construct Fat-tree and mesh topologies in which virtual switches and hosts are permitted to exchange traffic in real-life conditions. In the last layer, the monitoring, logging and storage layer gathers system statistics, resource consumption, performance indicators, are stored for analysis and comparison. These layers can be assembled in order to create a systematic and reproducible environment to evaluate the behavior of the various SDN controllers in a variety of network topologies.

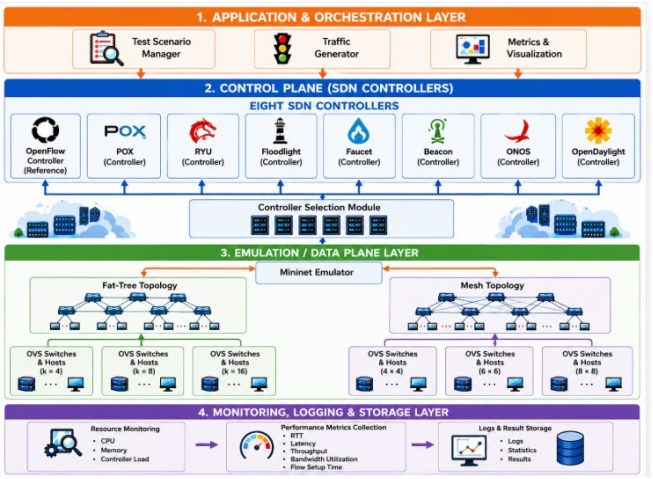

Figure 1. Proposed Architecture

3.1. Experimental Environment

All experiments are performed on a system with Intel Core i7 processor, 16 GB RAM running in Ubuntu (x86_64) using Mininet 2.3.0 and OpenFlow 1.3 protocol support with eight controllers. Topologies include Fat-tree with k value 8 and mesh with grid sizes 6×6. Each controller is connected to the topology emulated using the standard OpenFlow channel bandwidth settings of 10 Gbps, with traffic generation performed using iPerf over 60-second intervals and a packet

size of 1500 bytes. Measurements were repeated for the same traffic and emulation conditions for each controller-topology combination and the average values obtained in the tables were derived from the repeated observations. In order to ensure a fair comparison, the same set of hardware, operating-system, Mininet, OpenFlow, and traffic configurations were adopted during the experiments.

3.2. Performance Metrics

By using the following metrics, we evaluated the performance of the controllers.
Round-Trip Time (RTT) is the average time for a packet to travel from source to destination and back, measured using ICMP echo requests over multiple iterations denoted in (1).

$$\text{RTT} = \text{Tprop} + \text{Ttrans} + \text{Tqueue} + \text{Tproc} \quad (1)$$

Latency means end-to-end delay, including processing and queuing, computed from RTT and application-level measurements. Latency derived from RTT shown in (2).

$$Latency = \left(\frac{RTT}{2}\right) \quad (2)$$

Throughput is the amount of successfully delivered data per unit time (Mb/s or Gb/s) under different traffic loads. Its efficiency denoted in (3).

$$Throughput = \left(\frac{Tmax}{Tmeasured}\right) * 100 \quad (3)$$

Bandwidth utilization means effective use of available link capacity, reflecting how well the controller and forwarding rules exploit the Fat-Tree/Mesh structure.
The controller flow-setup time represents the total delay incurred when a switch forwards a new flow request to the controller, the controller processes it, and the resulting rule is returned to the switch expressed in (4).

$$Tflow - setup = Tswitch \rightarrow ctrl + Tctrlproc + Tctrl \rightarrow switch \quad (4)$$

Fat-Tree bisection bandwidth model is defined as the bisection bandwidth of a k-ary Fat-Tree is given by the total capacity available across the network's central aggregation stage shown in (5).

$$Bbisection = \left(\frac{k3}{4}\right).C \quad (5)$$

Mesh topology shortest path model is the hop distance between two nodes in a mesh grid is determined by the Manhattan metric denoted in (6).

$$dmesh(i,j) = |xi - xj| + |yi - yj| \quad (6)$$

Controller load model is the instantaneous load on a controller at time $t$is defined as the ratio between the number of flow requests it receives and its service capacity represented in (7).

$$L(t) = \frac{F(t)}{c} \quad (7)$$

Where, F(t)= number of flow requests, C= controller capacity

3.3. Experimental Procedure

The evaluation procedure followed a consistent sequence for every controller and topology under study. Each experiment started with launching Mininet, using Fat-tree or mesh topology. The layout was designed and then the switches were connected to the selected controller for the experiment purpose. After the environment was activated, the controller was enabled to finish the topology discovery process based on LLDP exchanges [19]. To evaluate the performance of controllers under the load, traffic tests were conducted and tested with delayed probes sent over ICMP and throughput probes sent out over iperf between different host pairs. Each performance value was recorded several times to ensure a good minimum, maximum, and average set of statistics. The same process was repeated for networks of different sizes by adjusting the Fat-Tree parameter k or the mesh grid dimension, providing a good idea of the scaling of each controller as the topology increases. In the next section we analysed the results by indicating the performance of various controllers.

4. Results and Analysis

4.1. Fat-Tree Topology: Controller Performance

As in Table 1, shows considering of Fat-Tree (k = 8) topology, ONOS and ODL are the best controllers. The four metrics measured like RTT, latency, throughput and bandwidth utilization.

Table 1. Fat-Tree Topology (k = 8) RTT, Latency, Throughput and Bandwidth

Controller	RTT Min (ms)	RTT Max (ms)	RTT Avg (ms)	Avg Latency(ms)	Throughput (Gb/s)	Bandwidth Utilization (%)
OpenFlowController	5.8	12.9	9.1	4.6	5.1	58%
POX	6.1	14.2	9.7	4.9	4.8	54%
RYU	4.9	10.1	7.4	3.7	6.2	70%
Floodlight	4.5	9.6	6.8	3.4	6.9	75%
ONOS	3.2	7.4	4.1	2.0	9.6	92%
OpenDaylight	3.4	7.9	4.4	2.2	9.2	89%
Faucet	4.8	10.4	7.1	3.6	6.5	72%
Beacon	4.2	9.3	6.2	3.1	7.1	78%

In terms of latency and RTT, ONOS achieves the lowest average RTT is 4.1 ms and lowest average latency is 2.0 ms

among all controllers. ODL closely follows with an average RTT of 4.4 ms and a latency of 2.2 ms. Both are clearly ahead of POX and the reference OpenFlow Controller, whose average RTT values are around 9 ms to10 ms, almost double that of ONOS. If we consider the throughput and bandwidth utilization of ONOS, it reaches the highest throughput at 9.6 Gb/s with 92% bandwidth utilization, while ODL achieves 9.2 Gb/s and 89% utilization. This shows that the both controllers are able to fully exploit the path diversity of Fat-tree fabric. In contrast, POX and the reference OpenFlow Controller remain below 5.1 Gb/s throughput and under 60% utilization, indicating that they do not use the available links efficiently. Floodlight has a reasonable compromise with a 6.8 ms mean RTT, and latency of 3.4 ms, and throughput of 6.9 Gb/s (75% utilization). Beacon additionally has a good throughput (7.1 Gb/s, 78% utilization) and shows moderate latency. RYU and Faucet are good, although the throughput remains inferior to ONOS and ODL.

4.2. Mesh Topology: Controller Performance

The 6×6 mesh topology results indicate the performance differences between the controllers, particularly at the point where the network is denser and has several redundant paths.

Table 2. Mesh Topology — RTT, Latency, Throughput and Bandwidth

Controller	RTT Min (ms)	RTT Max (ms)	RTT Avg (ms)	Avg Latency (ms)	Throughput (Gb/s)	Bandwidth Utilization (%)
OpenFlowController	7.2	18.3	11.9	5.9	3.6	42%
POX	7.8	19.1	12.4	6.2	3.4	40%
RYU	6.1	14.2	9.3	4.6	4.9	55%
Floodlight	5.6	12.6	8.8	4.3	5.3	60%
ONOS	4.2	9.3	5.1	2.5	7.8	82%
OpenDaylight	4.4	9.7	5.6	2.8	7.5	80%
Faucet	6.4	13.1	9.7	4.8	5.1	58%
Beacon	5.2	11.4	7.9	3.9	5.7	63%

From Table 2, most notable similarity is that ONOS and ODL have lowest average values of RTT, which are 5.1 ms and 5.6 ms, respectively. The latency values of 2.5 ms (ONOS) and 2.8 ms (ODL) are lower than any other controllers. This shows that both controllers are responsive to mesh connectivity, in which there are many alternative paths that should be calculated, and administered effectively. In terms of high throughput and bandwidth, ONOS has the best throughput of 7.8 Gb/s at 82% bandwidth utilization, demonstrating its great ability to allocate traffic among available crosslinks of mesh. ODL is close behind at 7.5 Gb/s and 80% utilization. These two controllers observably utilise the path diversity of the mesh structure and result in the network being well-balanced, even when loaded more heavily. Here moderate performance controllers are floodlight, beacon, and RYU. Floodlight performs 5.3 Gb/s under 60% utilization, which is significantly smaller than ONOS and ODL. Beacon is better than floodlight in terms of delay and reaches 5.7 Gb/s, yet still unable to compete with the best two controllers. RYU achieves 9.3 ms RTT and 4.9 Gb/s throughput, which is suited for medium scale deployments. POX and OpenFlow reference controller show the highest delays and the lowest throughput. Hence, they have low end performance. Their average RTT values exceed 11 ms, and their bandwidth usage remains at 40–42%, indicating that they are not able to fully utilize the connectivity available in a mesh topology.

4.3. Throughput Comparison of Fat-Tree and Mesh Topologies

Table 3 represents the Fat-tree topology, ONOS reaches 9.6 Gb/s and ODL reaches 9.2 Gb/s, which are close to full network capacity. Even when moving to the more challenging 6×6 mesh topology, they still maintain strong performance with 7.8 Gb/s (ONOS) and 7.5 Gb/s (ODL).

Table 3. Throughput of Controllers in Fat-Tree (k=8) and 6×6 Mesh Topology

Controller	Fat-Tree Throughput (Gb/s)	Mesh Throughput (Gb/s)
OpenFlow Controller	5.1	3.6
POX	4.8	3.4
RYU	6.2	4.9
Floodlight	6.9	5.3
ONOS	9.6	7.8
OpenDaylight	9.2	7.5
Faucet	6.5	5.1
Beacon	7.1	5.7

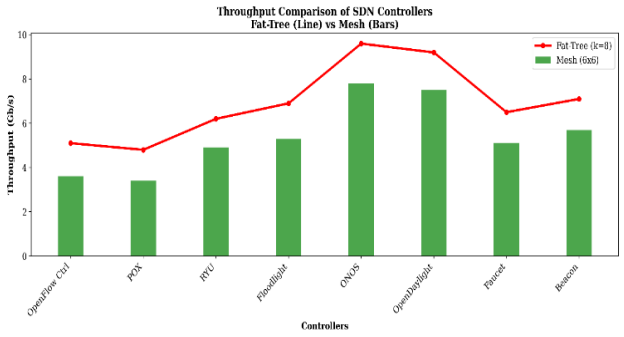

Figure 2. Throughput comparison of various SDN controllers: Fat-tree vs Mesh topology

The comparison of throughput is shown in Figure 2 and observed that ONOS and ODL performs better than remaining controllers.

4.4. Latency Values of Fat-tree and Mesh for all Controllers

In Table 4, the latency comparison between Fat-tree and Mesh topologies were presented. All controllers have a slight delay increase with transitioning to the more connected mesh network structure that inherently brings about longer and more diffuse paths.

Table 4. Latency Comparison of Fat-Tree and Mesh Topologies

Controller	Fat-Tree Avg Latency (ms)	Mesh Avg Latency (ms)
OpenFlow Controller	4.6	5.9
POX	4.9	6.2
RYU	3.7	4.6
Floodlight	3.4	4.3
ONOS	2.0	2.5
OpenDaylight	2.2	2.8
Faucet	3.6	4.8
Beacon	3.1	3.9

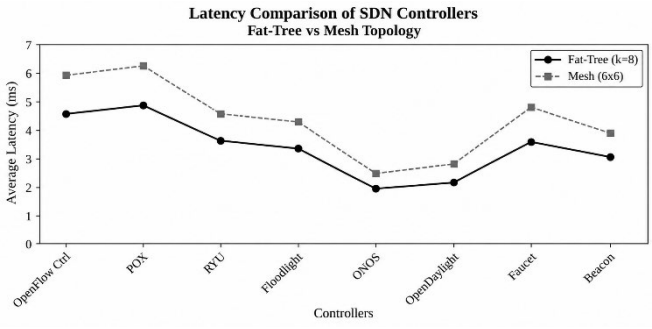

Figure 3. Latency comparison of various SDN controllers, Fat-tree vs Mesh topology

ONOS and ODL have the least latency values consistently at 2.0 ms and 2.2 ms, and only slightly increased to 2.5 ms and 2.8 ms in Fat-tree and mesh respectively as in Figure 3. The other performance layer is floodlight and RYU, which exhibit moderate latency in both topologies. Faucet and beacon also performing reasonably but with a marginally higher delay variation. In both environments, the OpenFlow reference controller and POX have the highest latency at 5.9 ms and 6.2 ms in mesh topology, implies they are not scalable. Both findings support the fact that the distributed and optimized controllers like ONOS and OpenDaylight are more efficient in dealing with complex topology changes and restore a stable and low-latency performance in both network topologies.

4.5. RTT of Fat-Tree and mesh for All Controllers

The comparison between RTT in Fat-tree and mesh topology shows the differences in terms of RTT controller responsiveness with the increase of network complexity. In both topologies, ONOS and ODL have the lowest RTT all the time. In ONOS, an average value RTT of 4.1 ms is recorded in Fat-tree, and 5.1 ms in mesh, and then ODL records 4.4 ms and 5.6 ms as shown in Figure 4.

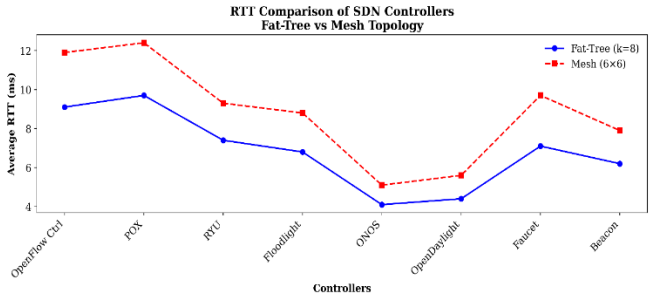

Figure 4. RTT comparison of various SDN controllers, Fat-tree vs Mesh Topology

These outcomes represent the performance of their distributed control-plane architectures, which are useful in ensuring constant performance despite the path, diversity and more routing decisions are utilised. Floodlight, beacon, and RYU have moderate RTT with fair adaptability and still lag behind ONOS and ODL in structured and dense networks. In OpenFlow reference the maximum RTT values in both environments are recorded by the controller. POX struggle to adapt with large-scale topologies. The findings prove that ONOS and OpenDaylight scale much more and has maintained low RTT in hierarchical and intense inter-networks.

5. Cross-Topology Summary & Practical Implications for SDN Applications

The best performance of ONOS and ODL across both Fat-tree and mesh topologies explained well based on architectural decisions underlying their control frameworks [20, 21]. The controller state is replicated across multiple nodes, following a distributed control model, using a combination of raft and atomix clustering technologies in ONOS. This design provides improved RTT and throughput. Rather, ODL is based on a modular architecture based on a key feature of Model-Driven Service Abstraction Layer (MD-SAL) that separates network services from southbound protocols. MD-SAL optimizes event handling. It allows for asynchronous processing and minimizes topology updates and packet message overhead. Both controllers make use of parallel execution path, computation of flow and translation of rules. They also include built-in event aggregation functionality, which further eliminates duplicate control traffic. These architectural strengths can be attributed as the network grows and the connectivity increases. The results obtained from this study are quite similar to the needs of the present SDN based environment, and especially large scale and latency sensitive applications [22]. ONOS distributed control plane is appropriate for 5G core deployments, for example, to steer User Plane Function (UPF), to slice

networks, and to handle fast mobility events. This requires low latency decision making and quick path recalculation and highly supports multi-domain orchestration and service-chaining. Apart from this, ODL also has a strong support for multi-domain orchestration and service-chaining. Operators with heterogeneous networks across several sites or administrative regions can rely on frameworks for a solid foundation. ONOS/ODL allows the use of this topology to efficiently leverage path diversity and achieve high bisection bandwidth and congestion-free routing during large east-west traffic flows, which is a key characteristic of the Fat-tree topology used in hyperscale data centers. Both mesh topologies benefit from rapid failure recovery and even topology-state dissemination that occurs in the carrier transport and inter data center backbone network controllers [23]. These features emphasise the usefulness of the proposed evaluation in discussing the practical utility of controller behavior observed in the experimental conditions directly maps to actual situations.

6. Conclusion

This work compared eight OpenFlow controllers on two realistic large scale topologies Fat-tree and mesh to know how these controllers could be used in the cloud, data-center, and 5G core topologies. However, from the results of this experiment, it is evident that ONOS and ODL can be considerably different. ONOS is more scalable and efficient compared to the other controllers that were reviewed, with a mean RTT of 4.1ms, latency of 2.0ms, and the throughput 9.6 Gb/s and ODL got 4.4 ms and 9.2 Gb/s, which is better than other controllers**.** In mesh topologies, where failover and recomputation are more difficult, ONOS had a mean RTT of 5.1 ms and again the top in throughput with 7.8 Gb/s. ODL was close behind 5.6 ms RTT and 7.5 Gb/s throughput. RYU and floodlight controllers were moderate scalability, whereas POX and the reference OpenFlow controller had difficulties with growing network size and traffic intensity. These findings confirm that controller selection must align with both topology and scalability. The distributed controllers with robust clustering, such as ONOS and ODL are the most suitable for high-bandwidth, multi-path environments typical of hyperscale clouds and 5G transport networks. The limitations in this study include the consideration of only homogeneous Fat-tree and mesh configurations in the controlled emulation settings. Future research will address heterogeneous and hybrid topologies and investigate the placement and dynamic load-aware optimization of controllers in larger and more diverse SDN networks with AI assistance. This analysis will be expanded to energy-aware control-plane placement, real-time telemetry integration, and P4-enabled data planes, enhancing the position of SDN in carrier and cloud infrastructures of the next generation even further.

References

[1] Rademacher M, Jonas K, Siebertz F, Rzyska A, Schlebusch M, Kessel M. Software-defined wireless mesh networking: Current status and challenges. Comput J. 2017; 60:1–12.

[2] Prasanthi BV, Reddy PC. Software Defined Networking (SDN) Technologies and Architectures for Efficient, Adaptive Networks. In: Koganti KK, SR E, Gupta N, editors. Advanced Technologies in Electronics, Communications and Signal Processing. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 619. Cham: Springer; 2026. p. 27–38.

[3] Mittal S. Performance evaluation of OpenFlow SDN controllers. In: Proceedings of the 17th International Conference on Intelligent Systems Design and Applications (ISDA 2017); 2017; Delhi, India. Cham: Springer; 2017. p. 913–923.

[4] Ghalwash H, Huang CH. QoS for SDN-based fat-tree networks. In: Lecture Notes in Networks and Systems. Cham: Springer; 2020. p. 1–10.

[5] Adanza D, Gifre L, Alemany P, Fernández-Palacios JP, González-de-Dios O, Muñoz R, Vilalta R. Enabling traffic forecasting with cloud-native SDN controller in transport networks. Comput Netw. 2024; 250:110565.

[6] Ryait DK, Sharma M. Comparative analysis of SDN controllers. In: Advances in Networks, Intelligence and Computing. Boca Raton, FL, USA: CRC Press; 2024. p. 156–164.

[7] Gilani SSA, Qayyum A, Bin Rais RN, Bano M. SDNMesh: An SDN-based routing architecture for wireless mesh networks. IEEE Access. 2020; 8:1–15.

[8] Correa Chica JC, Imbachi JC, Botero Vega JF. Security in SDN: A comprehensive survey. J Netw Comput Appl. 2020; 159:102595.

[9] Abdi AH, Audah L, Salh A, Alhartomi MA, Rasheed H, Ahmed S, Tahir A. Security control and data planes of SDN: A comprehensive review of traditional, AI, and MTD approaches. IEEE Access. 2024; 12:1–25.

[10] Sharma A, Verma GS. Performance comparison of Ryu and Floodlight SDN controller. In: AIP Conference Proceedings; 2023. Melville, NY, USA: AIP Publishing; 2023. p. 1–8.

[11] Wang YC, You SY. An efficient route management framework for load balance and overhead reduction in SDN-based data center networks. IEEE Trans Netw Serv Manag. 2018;15(2):1–12.

[12] Dhanya RP, Anitha VS. Implementation and performance evaluation of load-balanced routing in SDN-based fat-tree data center. In: Proceedings of the 6th International Conference on Information Systems and Computer Networks (ISCON); 2023. Piscataway, NJ, USA: IEEE; 2023. p. 1–6.

[13] Arahunashi AK, Neethu S, Aradhya HVR. Performance analysis of various SDN controllers in Mininet emulator. In: Proceedings of the 4th International Conference on Recent Trends on Electronics, Information, Communication and Technology (RTEICT); 2019. Piscataway, NJ, USA: IEEE; 2019. p. 752–756.

[14] Bano M, Qayyum A, Bin Rais RN, Gilani SSA. Soft-Mesh: A robust routing architecture for hybrid SDN and wireless mesh networks. IEEE Access. 2021; 9:1–15.

[15] Alssaheli OMA, Abidin ZZ, Zakaria NA, Abas ZA. Implementation of network traffic monitoring using software-defined networking Ryu controller. WSEAS Trans Syst Control. 2021; 16:1–10.

[16] Bao K, Matyjas JD, Hu F, Kumar S. Intelligent software-defined mesh networks with link-failure adaptive traffic balancing. IEEE Trans Cogn Commun Netw. 2018;4(3):1–12.

[17] Yu F, Zeng X, Cheng L. Performance optimization and simulation of SDN fat-tree network based on Dijkstra's

algorithm. In: Proceedings of SPIE; 2022. Bellingham, WA, USA: SPIE; 2022. p. 1–8.
[18] Prasanthi BV, Reddy PC. Topology aware evaluation of SDN controller performance and placement strategies in star, linear, and ring networks. *Cybern Inf Technol.* 2026;26(2):233–251.
[19] Guo X, Wang C, Cao L, et al. A novel security mechanism for software defined network based on blockchain. Comput Sci Inf Syst. 2022;19(2):523–554.
[20] Ghalwash H, Huang CH. A congestion control mechanism for SDN-based fat-tree networks. In: Proceedings of the IEEE High-Performance Extreme Computing Conference (HPEC); 2020. Piscataway, NJ, USA: IEEE; 2020. p. 1–6.
[21] Magalhães E, Garrich M, Carvalho H, Magalhães M, González N, Oliveira J, Bordonalli A. Global WSS-based equalization strategies for SDN metropolitan mesh optical networks. In: Proceedings of the European Conference on Optical Communication (ECOC); 2014. Cannes, France: IEEE; 2014. p. 1–3.
[22] Kim, W. S., Chung, S. H., & Moon, J. W. (2015). Improved content management for information-centric networking in SDN-based wireless mesh network. *Computer Networks*, *92*, 316-329.
[23] Sgambelluri A, Giorgetti A, Cugini F, et al. OpenFlow-based segment protection in Ethernet networks. J Opt Commun Netw. 2013;5(9):1066–1075.