From Edge to Control: Benchmarking Software Defined Network Performance in IoT Networks

K Hamdi Mohamed G Alama[1], Suriani Mohd Sam[2*], Mohd Hizami Ab Halim[3], Norliza Mohamed[4], Hazilah Mad Kaidi[5]

[1,2,3,4,5]*Faculty of Artificial Intelligence, Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia*

Laama.Hamdi@outlook.com, suriani.kl@utm.my, mhabhalim@graduate.utm.my, norlizam.kl@utm.my, hazilah.kl@utm.my

Article history

Received:
2 April 2026

Received in revised form:
15 April 2026

Accepted:
27 April 2026

Published online:
15 June 2026

*Corresponding author
suriani.kl@utm.my

Abstract

Software Defined Networking (SDN) is an emerging technology that addresses networking complexity and management by separating the control plane from the data plane in switches and routers. This approach introduces key features such as centralized control and global network visibility, which support automation. In the context of the Internet of Things (IoT), SDN helps manage the wide variety of participating devices. By combining SDN controllers, Open vSwitch, and network virtualization, it offers significant operational and management benefits for IoT architectures. However, despite these advantages, existing studies have largely focused on conceptual models or simulated environments, providing limited empirical data on SDN controller performance within real IoT network conditions. This lack of experimental benchmarking makes it difficult to determine how different controllers behave under varying network loads and scales. Addressing this gap is essential to ensure reliable controller selection and deployment in IoT ecosystems. This study investigates SDN controller implementation in IoT networks, focusing on performance evaluation using two controllers, OpenDayLight and Floodlight. Performance is measured in terms of throughput and latency across three scenarios. Unlike prior comparative studies, this work provides a practical benchmarking framework using a real IoT testbed built on Raspberry Pi and Open vSwitch, offering new insights into controller behavior from the network edge to the control layer.

Keywords: *We would like to encourage you to list your five relevant keywords separated with a comma (,) in this section*

1. Introduction

In recent years, the nature of traditional networking architecture and network-connected devices has shifted dramatically. As a result, a closer examination of the traditional networking architecture and its various layers or parts may appear ideal. However, networkers have had to deal with the challenges of producing diverse content while efficiently deploying new devices and policies into an already existing

* Corresponding author. suriani.kl@utm.my

network infrastructure. Thus, the network redesign and expansion task faces a significant challenge that imposes high costs on network owners in several cases, such as the emergence of the Internet of Things (IoT).

The Internet of Things will be made up of these connected devices, which will most likely generate more massive amounts of data transmitted over the Internet. With this massive amount of data, traditional network architectures will struggle to keep up with the traction and magnitude of the data. For example, according to International Data Corporation (IDC) data, the amount of digital data generated in 2010 exceeded one zettabyte. Furthermore, according to another static analysis, 2.5 exabytes of new data have been generated every day since 2012. Furthermore, Cisco predicts that by 2020, approximately 50 billion devices will be connected to the internet (Evans, 2021). Thus, coordinators will be required regardless of the number of network functions or the cost of each network expansion (Javanmardi et al., 2023).

These changes have resulted in new network requirements that necessitate the use of emerging technologies, as well as the search for a better alternative to the existing traditional networking architecture. This search resulted in the development of a new networking technology known as Software Defined Networking (SDN) (Babiker et al., 2022) and (Gomes et al., 2024). SDN is a new and viable network architecture that enables network programming. SDN opens new opportunities for creating and delivering new services. The SDN architecture breaks vertical integration by separating the network's control plane from the data plane and its underlying switches and routers, which are used to forward network traffic (Zoraida et al. 2024) and (Amirah et al., 2021). The switches will then be reduced to being nothing more than network forwarding devices. Finally, the control logic is implemented in a logically centralized controller to restructure the policy of enforcement, network reconfiguration, and development to empower cloud and system administrators and managers.

This paper contributes to the growing body of research on network performance optimization by benchmarking Software Defined Networking (SDN) in Internet of Things (IoT) environments. Specifically, it (1) evaluates SDN performance across different network layers and control configurations, (2) compares the efficiency of SDN-based control with traditional networking in managing IoT traffic, and (3) proposes a benchmarking framework that quantifies latency, throughput, and packet loss from the edge to the control layer. By doing so, the study offers empirical insights into how SDN can enhance scalability, flexibility, and control responsiveness in IoT networks, providing valuable references for both researchers and network practitioners seeking efficient network management in large-scale IoT deployments. Its purpose is to respond quickly to changing business requirements and changes by deploying new network requirements to the controller and covering the actual requirements (Koulouras et al., 2022) and (Ariffin et al., 2022).

3.3cm on the left side, 3.65cm on the right, 2.03cm on the top, and 3.05cm on the bottom. Paper orientation in all pages should be in portrait style.

2. SOFTWARE DEFINED NETWORKING

SDN has recently gone viral and has been discussed in networking technology. It introduces numerous new benefits and features that enable the resolution of several critical issues in traditional IP networks. This approach has introduced many new features, such as one-point control of network flow and viewing the entire network simultaneously. Furthermore, it aids in network automation and better server/network utilization by decoupling control plane functionality from data plane functionality (forwarding devices) and replacing it with controller-programmed software (Ahmad & Mir; 2022).

SDN have recently seen tremendous growth and deployment in different types of networks. For example, they are being actively used in data center networks, wireless & IoT networks, wide area & cellular networks, and security and privacy of domains (Khorsandroo et al., 2021). SDN is offered to enhance the network functionalities by its programmability, which allows the network to be very flexible in new configurations and deploy new applications in the network cost-effectively. It was invented as a new networking framework that tends to bring various new capabilities and provides a solution to many of the complex challenges of legacy networks. This approach depends on splitting the functionality of the control plane from the data plane (forwarding devices) so that it can be directly programmed and controlled by a logically centralized controller. The control algorithm manages traffic flow and forwarding decisions imposed in the switches using an OpenFlow protocol (Khorsandroo et al., 2021).

The motivation of SDN is to operate as a network operating system, where network tasks can be done without including additional programmers for each switching element. It also offers the developers the chance to develop applications that can be used to control the switches by running them on top of the controller in the application layer (Oo, 2023). By deploying SDN controllers in IoT networks, data collection and management can be achieved efficiently through the SDN network virtualization function and the easy integration of new service applications at the SDN application layer.

SDN has five features contributing to network technology: (1) Network programmability. Network control can be programmed to perform a Citrine set of tasks based on the given operations instructions programmed on the controller. (2) Agile: extracting control of forwarding decisions from the network forwarding devices. It enables administrators to dynamically modify and adjust network traffic flow to adopt and meet any changes or expansions that might be needed within the network. (3) Centrally managed: This is achieved by having a centralized controller in charge of the entire network. (4) Programmable devices configuration: It provides easy configuration and network management through dynamic, automated SDN programs and the controllers' protocol. (5) Open standards-based and vendor-neutral: In deploying the SDN, the network design is more straightforward than traditional IP networks. The control or the network managing instruction are

handled by an open flow controller, not different vendors' devices, and protocols (Ayodele & Buttigieg; 2024) and (Javanmardi et al., 2023).

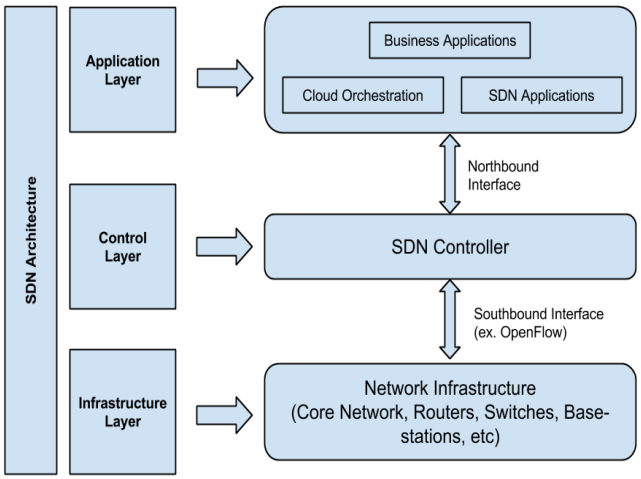

Fig. 1. Overview of Software-Defined Networking Architecture.

Fig.1 shows the SDN architecture. The architectures decouple the network control and forwarding plane. This action helps to control the network and program it directly. Direct programming builds the underlying network infrastructure and allows different applications and services on the network (Ayodele & Buttigieg; 2024). In the above diagram, the control plane and the data plane have been spilt. The control plane is now under the authority of a controller, which will then act as the network operating system. In an SDN, if the controller is turned off or goes down, the forwarding devices or other devices attached to the network controlled by the controller will stop working (Babiker et al., 2022).

In SDN, the controller is the software through which the network will be deployed, managed and maintained. It is crucial to ensure that the best controller is used for an SDN network because the controller determines the proper functioning of the network. The controller is a centralized entity managing and maintaining the network functions. It is also considered the brain of the SDN network, which is responsible for handling and forwarding the network traffic. Therefore, selecting the appropriate controller that provides the best performance to be deployed in your network is essential. Although there are many SDN controllers in the market, this study focuses on two types of controllers: OpenDayLight (ODL) Controller and Floodlight Controller.

ODL is another Open source controller for the SDN infrastructure that offers high availability, scalability and excellent fault tolerance; it is a java-based controller. ODL supports OpenFlow as well because it uses to control or communicate with the underlying network devices. In addition, it is the protocol that ODL uses in its Southbound API. Fig. 2 shows the architecture of the ODL Controller (Singh et al., 2022).

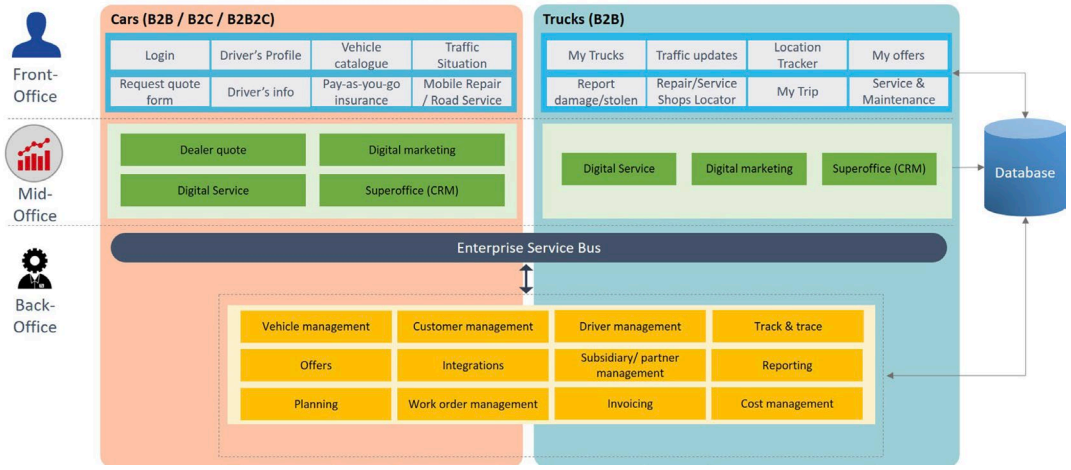

Fig. 2. The ODL Controller Architecture

The ODL supports open Northbound APIs, applications used through the controller to collect information concerning the network, and algorithms used to perform analytics. In addition, the controller will create new rules for the network to ensure better performance. Stanford University invented the Floodlight Controller. The controller promotes the Big Switch network's most famous open-source community contribution. It is licensed from Apache and includes a JavaScript service (non-OSGI). It can support the commercial enterprise in offering a big network controller (BNC) (Hui et al., 2025).

The concept of modular architecture is used to explain the view of floodlight controllers. The prominent architecture provides information about the modules, including the typology and device/end station management. It also includes the information related to the route of computation and the infrastructure used for accessing the web. The example of open flow counters and state storage systems is also included in the explanation, which are well stitched by the module-management system. Below in Fig. 3 are the essential components of the controller architecture.

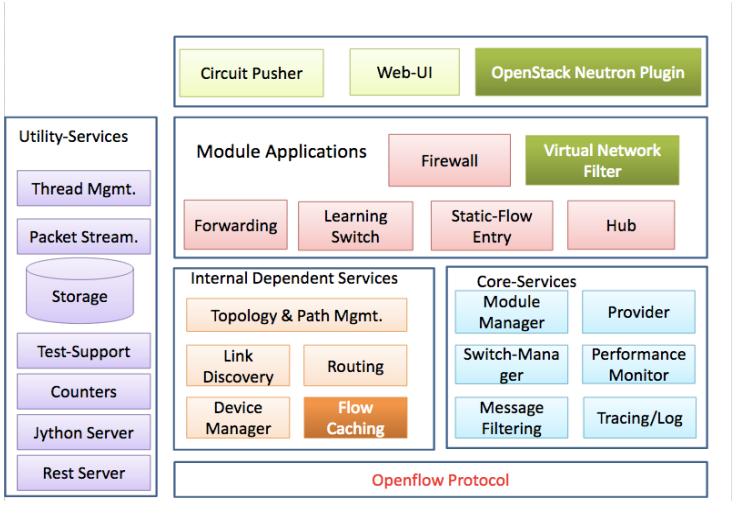

Fig. 3. The FloodLight Controller Architecture

Floodlight controllers have several applications that use the exposed REST APIs. For example, circuit Pusher uses floodlight rest APIs to create a route between two IP devices. It can be done with the help of flow entries in the switches which make up the entire route. Moreover, the Floodlight can be used to network on the base of the Neutron plugin as it bases on an Open stack. As initially outlined, SDN alludes to a mechanism wherever the forwarding state within the data plane is managed by a distant management plane decoupled from the previous. The networking industry has moved from this unique perspective of SDN by alluding to anything that includes programmable software as being SDN. SDN Controllers are in no doubt the brain of the networks which they are deployed in, therefore the performance of the network solely depends on the controller which makes it important for controllers to be tested before they are deployed to determine whether they can perform properly even in the most challenging situation as the overall networks of today are highly unpredictable (Villicaña, 2024).

As reported by (Singh et al., 2022), OpenDaylight (ODL) is compared with Floodlight, another Java-based controller, to evaluate relative capabilities.Here it was highlighted that the ODL Architecture is based on Services Abstraction Layer (SAL) concept, and ODL, as earlier stated, is an Open Source controller. It is supported by top networking vendors like Juniper and Cisco. It focuses on determining some significant and effective ways of preventing malicious attacks on SDN. The results of the implementation can record 10% false positive (FP) and counterfeit negative (FN) and 90% correct positive (TP) and correct negative (TN) for the shown algorithm.

In a focused analysis, (Haas, 2021) explores the key challenges and effects of software-defined networking. OpenFlow POX controllers and SDN Controllers were used to examine the challenges and effects of software-defined networking. The results indicated that the software portrayed networking is presented as an evolutionary paradigm shift. However, there are still experiences several issues. The result also showed that a precise absence of know-how, united with greater complexity when integrating into traditional networks, is the crucial reason for the delayed diffusion of the technology. (Khorsandroo et al., 2021) has examined the obtainable knowledge concerning SDN through a systematic literature review (SLR) to assess the current SDN situation of the art based on research tracks as well as publications and trends using the OpenFlow POX controller and the OpenFlow-based recursive SDN controller. The results outline data concerning the most active research sectors, tools, security problems, obstacles, limits, and strengths and opportunities in SDN.

On the other hand, (Amirah et al., 2021) experimented using Raspberry Pi boards. SDN is conducted to evaluate the performance of SDN architectures and deployment based on an experiment using Floodlight controller and Open vSwitch. Authors in (Koulouras et al., 2022) and (Straesser et al., 2023) extensively researched the scalability and performance of SDN controllers based on throughput, latency and network topology discovery time. Their purpose was to look for the present scenarios. The research is done for NOX, POX, Floodlight, ODL, ONOS, Ryu, OpenMUL, Beacon, and Maestro network controllers.

These papers have provided a comprehensive qualitative comparison of different SDN controllers and a quantitative analysis of their performance in different network scenarios. Also, it provided a depth capabilities comparison of three benchmarking tools used for SDN controllers, along with best practices for quantitative controller evaluation. This project aims to evaluate the performance of the SDN controllers in IoT networks using Raspberry Pi 4, the most recent version of Raspberry Pi.

3. METHODOLOGY

The operational design of this project consists of several steps. First, the operational design includes a structured set of planned activities to achieve the objectives. Besides, it is also used for monitoring the project progress in each stage. In order to achieve the objectives of this project and obtain a positive result, a methodology involves the following stages: Planning, Designing, Implementing, Testing, Operation and analysis and documentation (Jaadouni et al., 2022). Each stage was systematically executed to ensure that the experimental setup was consistent, reproducible, and accurately reflected real-world network performance. Fig. 4 illustrates the operational design processes.

Phase 1: Planning

The planning stage involved defining the research problem, identifying performance parameters, and formulating the experimental objectives. The study aimed to benchmark SDN controllers in IoT networks by comparing throughput and latency performance. A literature review was conducted to analyze existing approaches to SDN-IoT integration and to identify performance gaps in previous studies. The controllers were selected based on their open-source nature, stability, and support for OpenFlow—namely OpenDayLight (ODL) and Floodlight. The planning phase also included determining network depth configurations and developing a conceptual model linking the edge layer (IoT devices) to the SDN control layer.

Phase 2: Designing

During the design phase, the SDN-IoT testbed architecture was created to simulate real IoT networking conditions. A Raspberry Pi 4 Model B was configured as an Open vSwitch (OVS) to act as a network bridge between hosts and controllers. Two Virtual Machines (VMs) were created using Oracle VirtualBox, with each VM hosting one SDN controller (ODL and Floodlight). Mininet (v2.3.0) was used to emulate three distinct network depths: S1, S2, and S3. These represented small, medium, and large-scale IoT topologies using a tree structure. Each topology was assigned specific hosts and switches to evaluate scalability and control responsiveness. The Iperf3 tool was selected for throughput testing, while latency measurement relied on time-based data collection through Mininet's monitoring utilities.

Phase 3: Implementation

The implementation phase focused on deploying the designed network. Raspbian OS was installed on the Raspberry Pi, which was then configured with Open vSwitch (version 2.13.1). The Raspberry Pi was connected to the external controller-hosting VMs through a bridge interface (br0), enabling packet forwarding and control communication. OpenDayLight and Floodlight controllers were installed and configured to communicate with the OVS via TCP connections on specific ports. Network configurations, such as flow tables, IP addressing, and routing rules, were defined to ensure stable operation. Mininet was then used to instantiate virtual hosts and switches representing IoT nodes. This environment replicated real-time network operations between the data and control planes.

Phase 4: Testing and Operation

Testing was conducted to evaluate the throughput and latency performance of both controllers under identical network conditions. For each scenario (S1, S2, and S3), throughput tests were performed using Iperf3 in both TCP and UDP modes. Each test ran for 15 seconds with 10 MB file transfers between client and server hosts. The tests were repeated five times to calculate average throughput and identify consistent performance patterns. Latency was measured across fifteen iterations (T3–T15) to observe stability trends and controller responsiveness. The recorded values captured transmission delays, packet losses, and control response times. All results were logged for comparison and validation. Network traffic flows were continuously monitored to ensure that the test conditions were identical for both controllers (Ayodele & Buttigieg; 2024; Paramasivam & Velusamy; 2023).

Phase 5: Analysis and Documentation

In the analysis phase, collected data from throughput and latency tests were processed and visualized. The results were analyzed quantitatively to determine average, maximum, and minimum values for each metric under all network depths. Comparative analysis was carried out to assess the efficiency, scalability, and responsiveness of ODL and Floodlight. The findings were plotted using Gnuplot (v5.2) to visualize controller performance trends across varying IoT network sizes. This phase also involved compiling technical documentation of configurations, observations, and experimental outputs. Finally, recommendations were made for controller selection in future SDN-IoT deployments based on performance behavior.

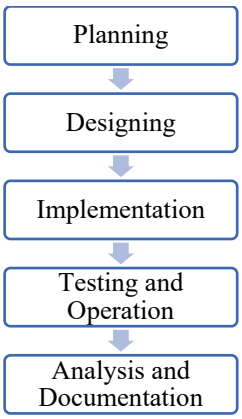

Fig.4. Flow Chart of SDN–IoT Process

SDN operational mechanism and packet handling are illustrated below in Fig. 5. SDN operations begin with the entry of packets into the network system. Testing will be done on the packet to ensure packet matching against the flow table. If packet matching is found, SDN will perform packet action, increase the counter, and match the header field. If not matching, SDN will notify the controller with a packet-in-message for processing. If there is a packet-in-message, SDN will send out a packet-out message with a new flow entry for the packet if no packet is discarded. This operation will continue as long as new packets are in the network system.

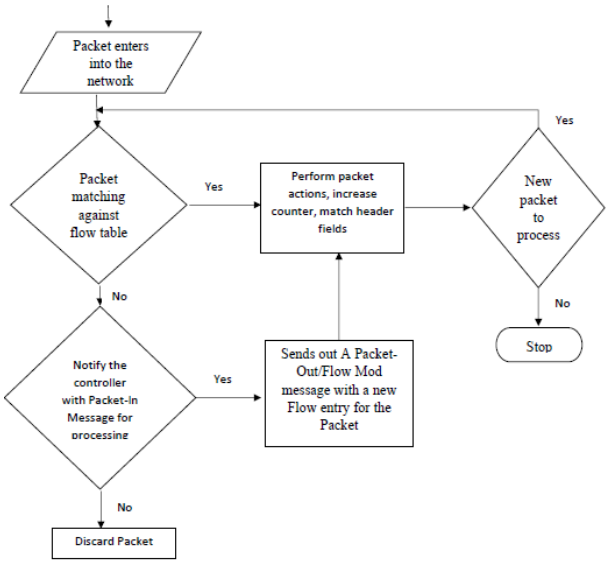

Fig.5. SDN operational mechanism and packet handling

3.1 HARDWARE AND SOFTWARE REQUIREMENTS

This study utilized both hardware and software components to develop a cost-effective yet realistic SDN-IoT benchmarking environment. The performance evaluation of the controllers and implementation were conducted using several tools

and platforms set and configured accordingly to meet the designed methodology and implementation, as shown in Table 1 below.

The Raspberry Pi 4 Model B served as the IoT edge device and Open vSwitch platform, enabling traffic forwarding and flow control between virtual hosts and controllers. Oracle VirtualBox hosted two VMs, each configured with a specific controller, OpenDayLight and Floodlight. Ubuntu Desktop 20.10 provided the base operating system for the controller environment. Mininet simulated IoT network topologies with varying numbers of switches and hosts. Open vSwitch (v2.13.1) bridged the Raspberry Pi with the virtualized controllers to enable dynamic flow management.

For performance testing, Iperf3 was employed to generate TCP and UDP traffic streams, while Gnuplot (v5.2) was used to generate visual graphs of throughput and latency results. Remote configuration and monitoring were enabled through VNC Viewer (v6.13) and X11vnc, allowing the testbed to be managed securely from an external workstation. All components were interconnected through the bridge interface (br0) to form a unified SDN-IoT network environment.

These tools and configurations provided a reliable experimental foundation to evaluate SDN controller performance in terms of throughput and latency, ensuring that the benchmarking outcomes reflected real-world IoT networking conditions.

Table 1. Tools and Software's Integrated

No.	Tool	Type	Version
1.	Raspberry Pi	Microcomputer	4 B
2.	Oracle Virtual Box	Virtual machine hosting Linux OS	Version 6.0.14
3.	Ubuntu desktop	Linux OS	20.10 RPi
4.	Mininet	SDN simulator	Version 2.3.0
5.	VNC RealVNC viewer	Secure remote access software	6.13
6.	X11 vnc server	Remote access software Linux	X11VNC
7.	OpenVswitch	Virtual Switch	2.13.1
8.	OpendayLight	SDN controller	12.2 (Magnesium)
9.	Floodlight	SDN controller	2.1.1
10.	Iperf	Testing tool	Latest Iperf 3
11.	gnuplot	Graphing utility for Linux	Version 5.2

While preparing the research environment, the controllers were installed and set on an external VM outside the Raspberry Pi. The pi has been configured as Open vSwitch, enabling a network bridge between the switch and the individual controller machine. All the installations and configurations of the project components toward setting up the Raspberry Pi as an Open vSwitch and bridging to the external SDN controller. Furthermore, ODL and Floodlight SDN controllers have been installed and configured on separate Virtual Machines. Finally, the SDN controller was connected to the Open vSwitch(Raspberry Pi) through br0 to enable the connection between the controller and the switch flow table and carry out the evaluation process.

4. RESULTS AND DISCUSSION

As it has been mentioned earlier, two SDN controllers have been deployed in separate VMs and designed to connect with the Open vSwitch on the raspberry pi via the particular bridge. Additionally, for a more enhanced SDN controller experiment, MiNiNet is installed on the raspberry pi switch to virtualize network topology and create virtual host machines as OpenFlow nodes connected to the Open vSwitch installed. In order to test the network and the SDN controller associated, we will initiate sudo mn targeting the specified controller network IP address and listening to the port number of the particular controller.

First, initiate network topology and host connections through the Open vSwitch, and the controller has been added successfully. Rather than the connection-specific IP and port number required to initiate the network, particular network topology will be used to simulate the required network accordingly. Tree topology is one of the network topologies available with MiNiNet. Therefore, it can be customized accordingly to the network size preferred. Below are the network topologies metrics used in the project. Table 2 shows the scenarios and network of depth 3, which has three switch layers and the hosts' bottom layer.

Table 2. Scenarios and network Depth.

Depth	Number of switches	Number of hosts
1	1	2
2	3	4
3	7	8

Each depth represents one network scenario with different network size, as shown in Fig. 6; the network topology of S3 is illustrated.

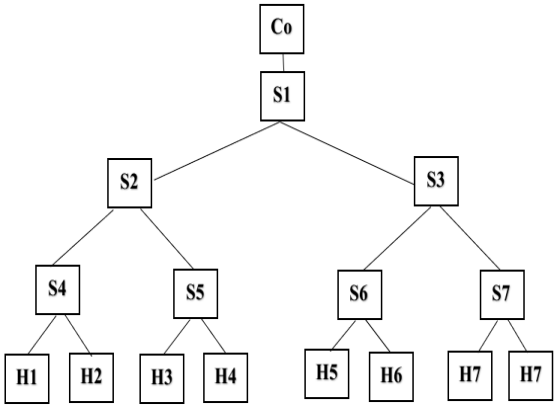

Fig. 6. Tree topology layout of depth 3(7S, 8H).

Testing and evaluation of the network have been conducted with two testing models to check the stability of the controller and the OpenFlow virtual hosts connected with the Open vSwitch. It has been done by testing the throughput-based client and server model of the two random hosts within the virtual network in both TCP/UDP flows.

The Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) throughput have been conducted for tests. Both protocols are critical to Internet operations. TCP ensures secure data transfer. On the other hand, UDP properties are speed and efficiency. where a three-way handshake is used to create a TCP connection. It is a process of making and recognizing connections. Once a connection has been made, data transfer can start. When it is finished, the connection is cut off by closing the formed virtual circuit. The UDP throughput rate is fixed up to the packet size sent and received within the Testing in the testbed network. UDP is the protocol that transfers data over IP and, most importantly, is not impacted by latency. One of UDP's principles is that its throughput is calculated based on packets sent and received by the other party (Rashid et al., 2023). Therefore, since the packet transmitted is well received without any loss, it always gives a fixed rate regardless of the time delay.

Table 3. Testing Throughput overview table.

Network Scenario	Time in secs	Source	Destination	Network Depth
S1	15	H1Server	H2 Client	1s/2h
S2	15	H2-S	H4-C	2/4h
S3	15	H3-S	H5-C	3/8h

The testing and evaluation of the two controllers within the network have been carried out based on the three scenarios mentioned in Table 3 above. These scenarios represent the size of the network in terms of hosts and OFF switches connected in each testing scenario. The number of hosts and switches is increased from S1 to S2 to S3 to operate and evaluate the controllers in different network sizes and loads and to ensure average results accuracy.

The testing and evaluation phase results have been analyzed, as illustrated in Tables 2 and 3. The analysis compares the actual outputs of the evaluation process results of each controller with both parameters in different network scenarios. Finally, the results of the throughput and latency rate of the two SDN controllers-based network is calculated based on the average and maximum rate value for throughput in each network scenario.

4.1 OPENDAYLIGHT CONTROLLER EXECUTION

The OpenDayLight controller results using Open vSwitch and tree topology of 3 depths which will run five times for each depth and test to average results, especially for the throughput results. Throughput test for ODL controller is carried out in two ways to test the connection between the hosts as Client-server communication in TCP and UDP. Each time the test ran, iPerf was set to 15 sec and 15 flow communication.

Fig. 7 shows the results of the throughput test output of TCP/UDP connection between two host client and server connected to the OVS for S1. UDP tests are set with 10 MB file transfer for the client sever throughput testing and measurements. For UDP we clearly notice that UDP transfer rate is stabilize at 10.5 Mbps. The TCP protocol, on the other hand, starts with a low throughput value and then increases it after seven seconds. Then it returns a consistent result, although it is still low when compared to transactions using the UDP protocol.

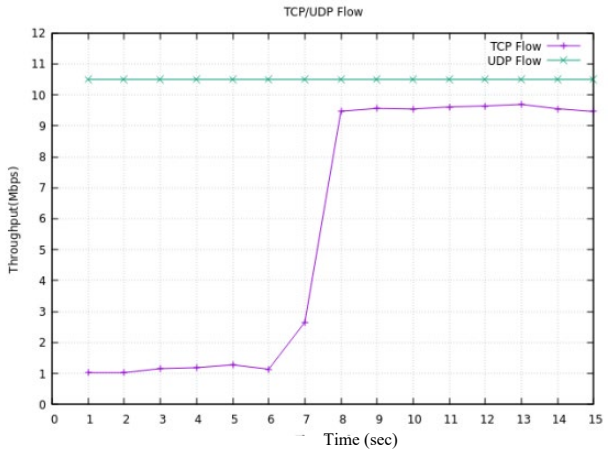

Fig. 7. The ODL TCP/UDP Throughput S1 Result

It was discovered that the TCP protocol had a higher throughput value at the depth of the two layer network. The throughput value for the S2 network is displayed in Fig. 8 as a result. Throughput figures between 7 Mbps and 8 Mbps are provided by communication outcomes utilising the TDP protocol using SDN from the ODL type via the IOT network and Raspberry PI. However, this value remains modest when compared to communication using the UDP protocol.

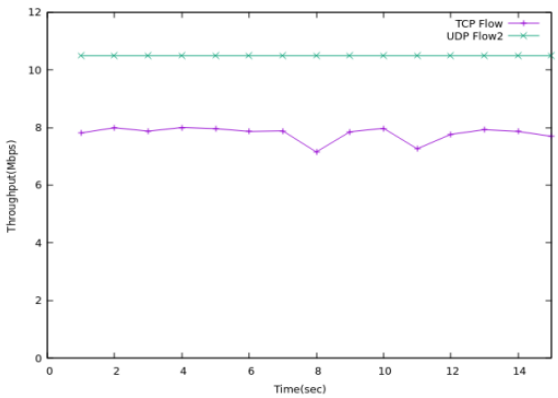

Fig. 8. The ODL TCP/UDP Throughput S2 Result

The network is then widened and its depth is increased to three layers as the study goes on. The throughput figures for the ODL Controller are shown in Fig. 9. It was discovered that the TCP protocol has a very good throughput value of more than 200Mbps. However, the UDP protocol's throughput figures continue to be the same as S1 and S2 result and are stable at 10.5 Mbps.

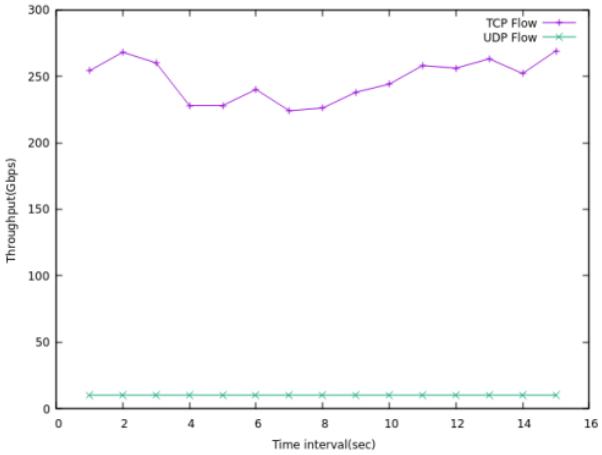

Fig. 9. The ODL TCP/UDP Throughput S3 Result

Fig. 10 displays the ODL controller's latency rate while using a Raspberry PI and an IoT network. All three types of network depth were found to have very low latency rates, with values of less than one second. However, it is apparent in this instance that the delay rate increases with the number of network levels. In Figure 10, the labels T3–T15 on the X-axis represent the test iterations or time intervals used to measure latency under different experimental conditions. Each test corresponds to a specific measurement instance where the network traffic was observed at varying conditions such as packet load, flow configuration, and network depth.

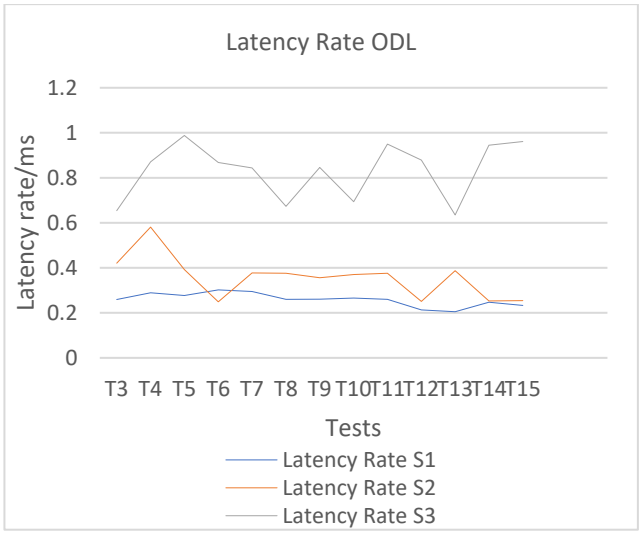

Fig. 10. The ODL Latency Result

4.2 FLOODLIGHT CONTROLLER EXECUTION

The floodlight controller results using Open vSwitch and tree topology of 3 depths which will run 15 times for each scenario of the networks. Throughput test for ODL controller is carried out based on the connection between the hosts as Client-server communication in TCP and UDP. Each time the test ran, iPerf was set to 15 sec and 15 flow communication.

Fig. 11, 12 and 13 displays the throughput value for TCP and UDP when using the Floodlight Controller. It was discovered that the TCP protocol's throughput value is less steady and less than the UDP protocol for all type of network. The UDP protocol has a throughput number that is more consistent and reliable, hovering around 10Gbps. Fig. 11 depicts the results of TCP and UDP protocol throughput values at the depth of the S1 network. The TCP protocol's throughput value was discovered to be in a fairly large range, ranging from 5 to 10 Gbps.

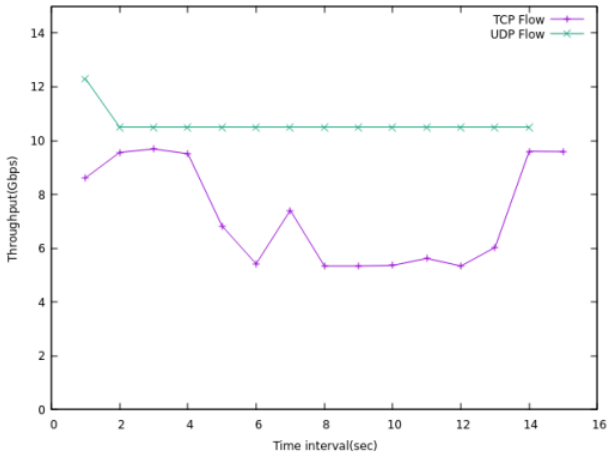

Fig. 11. Floodlight TCP/UDP Throughput S1 Result

Fig. 12 displays the throughput results for TCP and UDP protocol transactions with a two-layer network depth. Currently, the TCP protocol has a lower throughput value than UDP. However, the range of results is thinning to between 6 and 8 Gbps. The UDP throughput value remains constant at 10.5 GBps.

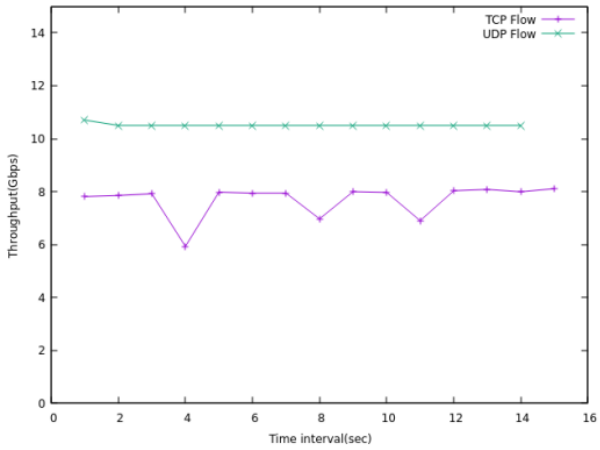

Fig. 12. Floodlight TCP/UDP Throughput S2 Result

Fig. 13 illustrates the throughput of TCP and UDP protocol transactions across three layers of a network. TCP protocol throughput currently remains low when compared to UDP. However, the range of results is narrowing, with results ranging from 6 to 7 Gbps. Throughput for UDP is continuously at 10.5 GB/s.

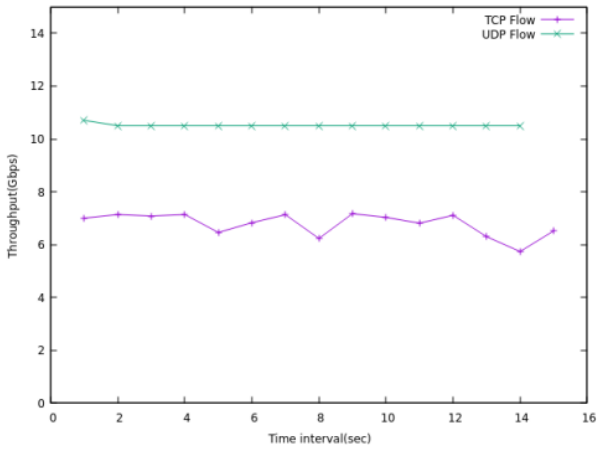

Fig. 13. Floodlight TCP/UDP Throughput S3 Result

Fig. 14 demonstrates the Floodlight SDN Controller's latency rate. The latency value was discovered to be very volatile for all three types of networks, with a relatively large range ranging from 0 to 14 seconds.

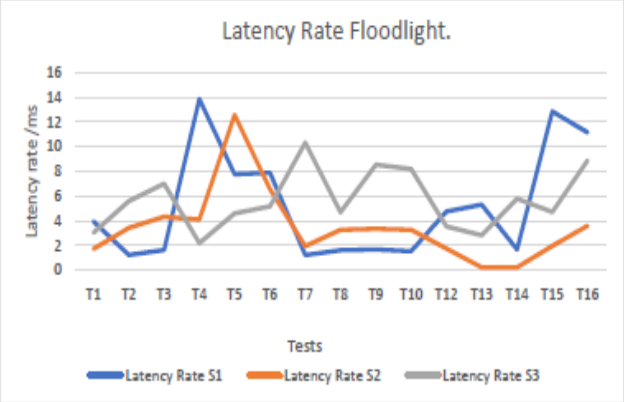

Fig. 14. Floodlight Latency Result

4.3 OPENDAYLIGHT CONTROLLER VS FLOODLIGHT CONTROLLER

A comparison between the controllers on the average results and the maximum rate achieved has been conducted. It is to compare and conclude the best of these controllers in the different scenarios based on the throughput and network latency.

A comparison of the throughput for the ODL controller and Floodlight controller is shown in Fig. 15. Floodlight throughput rate outperformed ODL in the case of S1 within small networks with significantly high deference in both the average and the maximum rate value. However, it is the opposite in S2 when the size of the network increases, where ODL achieve slightly better results. The result shows that S3 failed to achieve a reasonable rate, and some hosts.

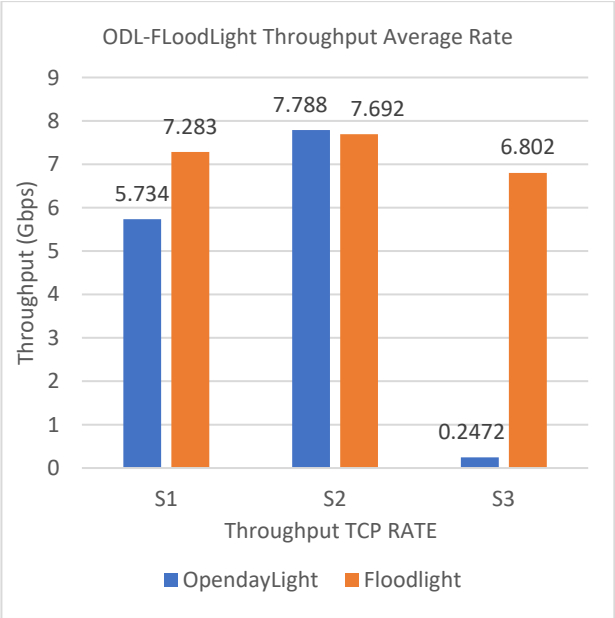

Fig. 15. ODL-Floodlight Throughput Average Rate.

Generally, delay always happens when the packet is transferred from one network entity to another. It can be defined as response time or round-trip time. However, latency is the sum of the time the packet from the source starts sending until the receiver completes receiving. Furthermore, the latency comparison of the two SDN-based networks has been considered based on the average results and the lowest latency value achieved for each network scenario set.

Fig. 16 compare the two controllers' average latency results and the lowest latency rate achieved. It determines which controller is better regarding the shortest/lowest network latency delay. As shown in figures below, ODL results are much better than the network latency results of the Floodlight controller in both cases and all network scenarios/sizes.

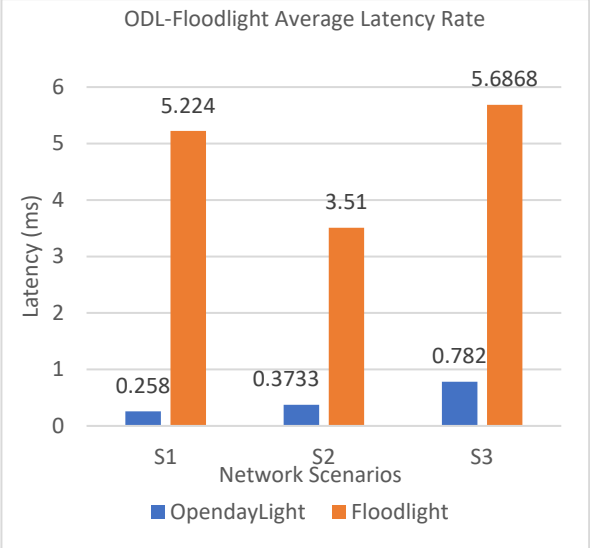

Fig. 16. ODL-Floodlight Latency Average Rate.

The study has briefly discussed the implementation of SDN controllers and Open vSwitch using Raspberry Pi as Open vSwitch and an OpenFlow-enabled switch. Therefore, the research is considered in a testbed, cost-effective research and evaluation environment. Additionally, at the end of the project, several aspects should be highlighted which is Implementation and configuration of OVS version 2.13.1, which have been set up and operated successfully with Raspberry Pi 4 model B. Implementation and deployment of two SDN controllers (OpenDaylight and Floodlight) were successfully within the designed network. We are successfully bridging between the controllers, the OVS, and other associated hosts' connections within the network. The evaluation of the designed network performance and the two selected SDN controllers is based on the network latency and throughput. This project has also identified the challenges of implementing SDN networking with different aspects of the Raspberry Pi and other integrated technologies.

Based on the results and analysis, it was evident that the Floodlight throughput rate is much better and outperforms ODL in several scenarios. Moreover, Floodlight is also more stable to operate in different network sizes. Yet, the case of S1 within small networks with significantly high deference in both the average and the maximum rate value. However, in terms of network latency, it can be noticed that ODL results are better compared to Floodlight. However, the latency rate of Floodlight would be okay in the actual network if it can achieve reasonable data rates regarding throughput.

5.0 CONCLUSIONS

In conclusion, the average and maximum value achieved for the throughput shows that Floodlight throughput rate results outperformed ODL. Furthermore, most of the scenarios considered Floodlight much more stable to operate well in the network than the OpenDayLight Controller, which failed to connect successfully to all switches and hosts. However, regarding network latency, it is the opposite, where OpenDayLight results were much better compared to the network latency result of the Floodlight controller. Although both controllers' results and the highest average value, which is 5.6 milliseconds. The two controllers are pretty good in latency tests and are to be deployed in actual networks. At the same time, the throughput rate is mainly the essential factor to determine. Finally, it can be concluded that the Floodlight controller is better to be deployed within the actual networks and dynamic networks where the network size may vary later, as this controller results in better throughput and is much more stable compared to OpenDayLight with several scenarios and network size.

ACKNOWLEDGEMENT

The authors fully acknowledge the Ministry of Education (MOE) and Universiti Teknologi Malaysia for the research grants Q.K130000.3556.05G32 that help in funding the research works.

Conflicts of Interest

The author declares that there is no conflict of interest regarding the publication of this paper.

REFERENCES

Ahmad, S., & Mir, A. H. (2022). SDN interfaces: Protocols, taxonomy and challenges. *International Journal of Wireless and Microwave Technologies*, 12(2), 11–32.

Ahmed, Z. E., Abd El-Latif, A. A., Riad, A. M., & El-Kassas, H. (2023). Mobility management enhancement in smart cities using software-defined networking. *Scientific African*, 22, e01932.

Amirah, N. F., & Ngah, L. (2021). Development of SDN controller testbed using Raspberry Pi 4. *International Journal of Synergy in Engineering and Technology*, 2(2), 29–38.

Ariffin, S. H. S., Chong, J. L., Abdul Latif, N. M., Nik Abd Malik, N. N., Arsad, S. S., Baharudin, M. A., Syed-Yusof, S. K., & Yusof, K. M. (2022). Intrusion detection system (IDS) accuracy testing for Software Defined Network Internet of Things (SDN-IoT) testbed. ELEKTRIKA: *Journal of Electrical Engineering*, 21(3), 23–27.

Arjona Villicaña, P. D. (2024). Testing Software Defined Networks with OpenDaylight and Mininet (Version 1.0). *Universidad Autónoma de San Luis Potosí.*

Ayodele, B., & Buttigieg, V. (2024). SDN as a defence mechanism: A comprehensive survey. International *Journal of Information Security*, 23, 141–185.

Babiker Mohamed, M., Alofe, O. M., Azad, M., Singh Lallie, H., Fatema, K., & Sharif, T. (2022). A comprehensive survey on secure software-defined network for the Internet of Things. *Transactions on Emerging Telecommunications Technologies*, 33(1), e4391.

Dave Evans. (2011, April). The Internet of Things: How the next evolution of the Internet is changing everything. Cisco Internet Business Solutions Group (IBSG), *Cisco White Paper*.

Gomes, J. E. C., Ehlert, R. R., Boesche, R. M., Santos de Lima, V., Stocchero, J. M., Barone, D. A. C., Wickboldt, J. A., Pignaton, E., Dos Anjos, J. C. S., & de Araujo Fernandes, R. Q. (2024). Surveying emerging network approaches for military command and control systems. *ACM Computing Surveys*, 56(6), Article 143, 1–38.

Haas, Z. J. (2021). Vulnerability challenges of software defined networking. *IEEE Communications Magazine*, 59(7), 88–93.

Hui, D., Hoh, W. S., Ong, B. L., Zhu, X., & Yoon, S.-K. (2025). Performance evaluation of Ryu, OpenDayLight and Floodlight controllers in diverse software-defined networking topologies. *Journal of Advanced Research Design,* 132(1), 103–114.

Jaadouni, H., Saadi, C., & Chaoui, H. (2022, November). Performance of OpenFlow-based SDN on IoT network. *IET Conference Proceedings*, 2022(14), 146–153.

Javanmardi, S., Shojafar, M., Mohammadi, R., Alazab, M., & Caruso, A. M. (2023). An SDN perspective IoT-Fog security: A survey. *Computer Networks*, 229, 1–19.

Keerthana, B., Balachandra, M., Hebbar, H., & Muniyal, B. (2022). Performance comparison of various controllers in different SDN topologies. In I. J. Jacob, F. M.

Gonzalez-Longatt, S. K. Shanmugam, & I. Izonin (Eds.), Expert Clouds and Applications – *Proceedings of ICOECA 2021 (Lecture Notes in Networks and Systems*, Vol. 209, pp. 297–309). Springer.

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

Koulouras, I., Margariti, S. V., Bobotsaris, I., Stergiou, E., & Stylios, C. (2022). 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.

Oo, B. B. (2023). Applied salt technique to secure steganographic algorithm. *Journal of Engineering Research and Sciences*, 2(2), 8–14.

Paramasivam, S., & Velusamy, R. L. (2023). Quality of service aware routing in software-defined video streaming: A survey. Peer-to-Peer Networking and Applications, 16(4), 1739–1760.

Rahouti, M., Xiong, K., Xin, Y., Jagatheesa Perumal, S. K., Ayyash, M., & Shaheed, M. (2022). SDN security review: Threat taxonomy, implications, and open challenges. *IEEE Access*, 10, 45820–45854.

Rashid, S. J., Alkababji, A. M., & Khidhir, A. S. M. (2023). Performance evaluation of software-defined networking controllers in wired and wireless networks. *TELKOMNIKA Telecommunication Computing Electronics and Control*, 21(1), 49–59.

Sajjad, M. M., Jayalath, D., & Bernardos, C. J. (2022). A flexible IPv6 mobility management architecture for SDN-based 5G mobile networks. *Transactions on Emerging Telecommunications Technologies*, 33(3), e3762.

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.

Straesser, M., Mathiasch, J., Bauer, A., & Kounev, S. (2023). A systematic approach for benchmarking of container orchestration frameworks. In *Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23)* (pp. 187–198). ACM.

Zoraida, B. S. E., & Indumathi, G. (2024). A comparative study on software-defined network with traditional networks. *TEM Journal*, 13(1), 167–176.