Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

An Efficacy of Multi-Controller Model Deployment Strategy in Software-Defined Networks

Virakwan Hai Kelian[1,*], Mohd Nazri Mohd Warip[1,3], R. Badlishah Ahmad[1,3], Phaklen Ehkan[1,3], Fazrul Faiz Zakaria[2,3], Mohd Zaizu Ilyas[1,3], Ng Yen Phing[4], Ali Ikhwan[5]

1 Advanced Computing, Centre of Excellence, Universiti Malaysia Perlis (UniMAP), Perlis, Malaysia
2 Micro System Technology, Centre of Excellence (CoE), Universiti Malaysia Perlis (UniMAP), Perlis, Malaysia
3 Faculty of Electronic Engineering Technology, Universiti Malaysia Perlis, Pauh Putra Campus, 02600 Arau, Perlis, Malaysia
4 Faculty of Computing and Information Technology (FOCS), Tunku Abdul Rahman University of Management and Technology (TARUMT)
5 Faculty of Science and Technology, University Islami Negeri Sumatera Utara, Medan, Indonesia

ABSTRACT

Software-defined networking (SDN) is being more widely acknowledged as a beneficial method for improving network flexibility and administration operations. The selection of controller architecture significantly impacts SDN performance. This study assesses the efficiency of centralized and multi-controller architectures in Software-Defined Networking (SDN) using the Mininet emulator and Ryu Controller for experimentation. Performance measures such as throughput, jitter, and memory utilization are examined. The study shows that the selection of controller design significantly impacts network performance. The centralized architecture showed increased throughput along with increasing memory usage. In some cases, the multi-controller equal role setup used less memory but had slightly higher jitter. The results emphasize the importance of carefully designing and optimizing controller architectures in SDNs to achieve specified performance objectives.

Keywords:
Software-defined networking; centralized controller architecture; multi-controller architecture; Ryu controller

1. Introduction

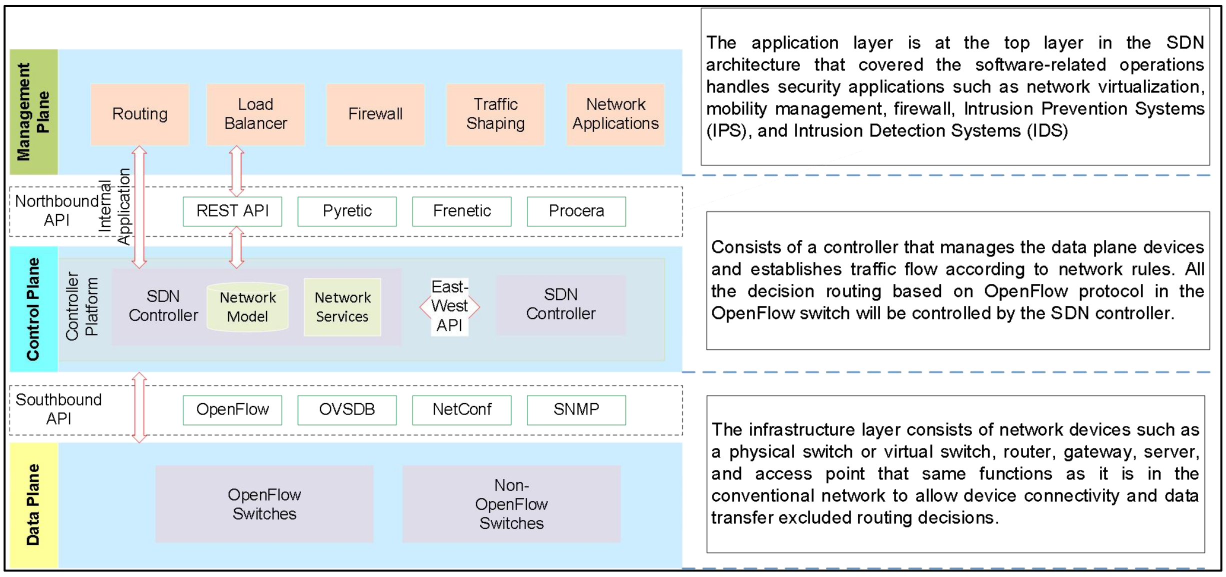
Software-defined networking (SDN) has emerged as a revolutionary paradigm in recent years. According to [1], SDN increases network flexibility, scalability, and management by separating the control plane from the data plane. SDN makes it possible to centralize the management and control of network resources, which makes dynamic network setups and the enforcement of policies easier. The controller architecture used in SDN is a crucial element that determines the operation of the control plane and its interaction with the network infrastructure, as referenced in [2]. The overall performance of SDN is substantially impacted by the selected controller architecture.

The architecture is the most essential aspect of the SDN concept since it can overcome the constraints of traditional networks through its design, as stated in [3]. Figure 1 displays how the

[*] *Corresponding author.*
E-mail address: virakwan@studentmail.unimap.edu.my

https://doi.org/10.37934/araset.57.2.146156

SDN architecture separates the network into separate layers of data plane, control plane, and management plane. There are two (2) fundamental features of SDN architecture. The first is the separation of the data and control planes, and the second is the integration of network intelligence into a centralized controller. A controller must have connectivity with all network elements to control the network.



The application layer is at the top layer in the SDN architecture that covered the software-related operations handles security applications such as network virtualization, mobility management, firewall, Intrusion Prevention Systems (IPS), and Intrusion Detection Systems (IDS)

Consists of a controller that manages the data plane devices and establishes traffic flow according to network rules. All the decision routing based on OpenFlow protocol in the OpenFlow switch will be controlled by the SDN controller.

The infrastructure layer consists of network devices such as a physical switch or virtual switch, router, gateway, server, and access point that same functions as it is in the conventional network to allow device connectivity and data transfer excluded routing decisions.

Fig. 1. Three-layer SDN architecture

1.1 Controller Architectures in Software-Defined Networks

Two well-known types of controller architecture are the centralized controller architecture and the distributed controller architecture, also known as the multi-controller architecture, as seen in Figure 2. The entire network is managed by a single controller in a centralized control architecture. As mentioned in [4], this controller is responsible for making global decisions and carrying out network policies. On the other hand, as noted by [5], distributed controller design enables localized decision-making and scalability by distributing control responsibilities across several controllers.

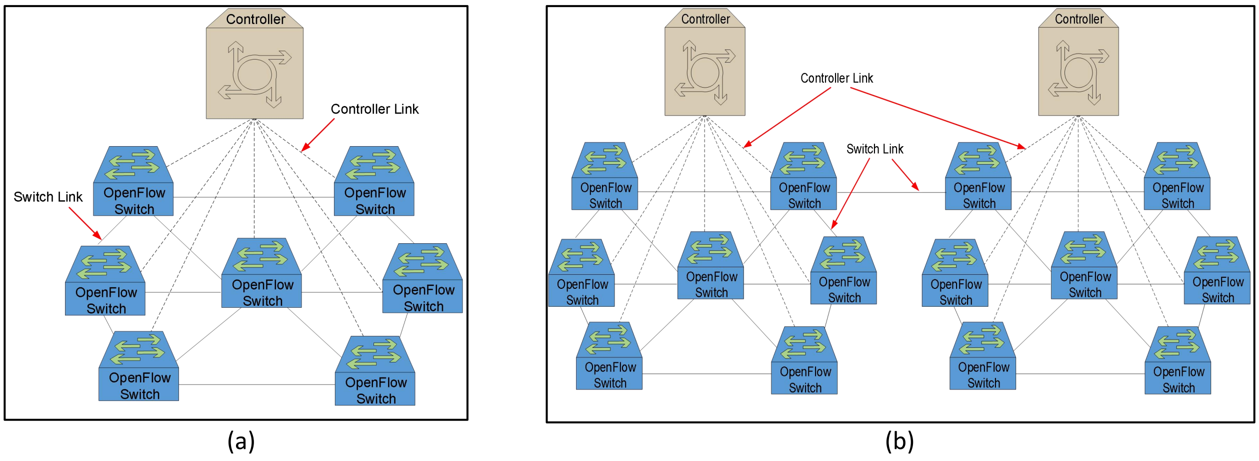


Fig. 2. SDN controller architecture (a) Centralized control plane (b) Distributed control plane

In connection with the existence of the type of controller architecture, numerous types of OpenFlow SDN controllers based on their control plane architecture are available. Most SDN controllers utilize the OpenFlow protocol to establish SDN communication standards for interaction between the controller and other networking devices, as referenced in [6]. NOX was the first OpenFlow controller released in early 2008, as mentioned in reference [7]. Initially, NOX was delivered in a single-threaded manner. Later in 2011, the current version of NOX was released, which is a C++ based controller that now enables a multithreaded learning switch application. In 2010, several alternative OpenFlow controllers were introduced. Beacon is a high-performance, multithreaded OpenFlow controller developed in Java as referenced in [8]. This controller can terminate both existing and new applications in real time as described in [9]. Ryu controller is a commonly used open-source SDN controller in research evaluations. The Python-based controller enables centralized control in Software-Defined Networks by supporting several OpenFlow versions. Ryu may be expanded to accommodate distributed controller designs and can be coded in the C programming language as specified in [10]. According to [11], developers can quickly design new network management and control applications by using Ryu's components, which provide excellent programming interfaces. Existing components can be readily updated and integrated into existing networks to meet the evolving requirements of applications that use these components. Ryu is a good alternative for small commercial and experimental applications as mentioned in references [7] and [10] due to its excellent characteristics.

1.2 Related Work

Software-defined networking (SDN) has garnered significant attention in the research community, leading to numerous studies investigating various aspects of control architectures in SDNs. This section reviews the related work that has examined the performance evaluation of control architectures in SDN, focusing on scalability and network performance. Numerous studies have evaluated the effectiveness of controller architectures in SDN. In earlier works, scalability evaluation has taken significant focus. The constraints of centralized controller architectures in large-scale networks have been studied, with a focus on the difficulties encountered when handling growing network size and traffic volume. Author in [12] examined the performance of various topological SDN networks, including bus, star, and tree, using a POX centralized controller. They observed that networks with a greater number of hosts had significantly lower performance, and the POX controller was incredibly simple to use and configure but struggled in complex networks with higher bandwidths.

The author in [7] assessed the performance of the parameters bandwidth, throughput, round-trip time, jitter, and packet loss between nodes. In this study, a Mininet emulator with a controller RYU with a switching hub component, one Open Flow switch, and three nodes is employed. Similar research was done by Bhardwaj and Panda [10] to examine the performance of the bandwidth, throughput, and roundtrip time for a four-host bus SDN network topology employing Ryu as the centralized controller. The primary focus of these investigations has been measuring network traffic analysis for node-to-node performance. The author of [13] compared the impact of heavy network traffic on centralized and distributed SDN architectures. The authors of this study implemented a Mininet emulator with the POX SDN controller to evaluate the average delay, maximal delay, average jitter, and average bitrate of complex topologies. In [14] and [15], a comparison was made between the effectiveness of centralized and distributed controllers using a load-balancing algorithm. A Mininet platform with a POX controller was used to simulate an SDN environment. According to the findings of this study, distributed SDN controllers are the best option for achieving higher performance and reducing the bottleneck, single point of failure, and scalability issues that plague centralized controllers. Lastly, the authors of [11] and [16] focused on the comparison of various types of centralized and distributed controllers. The authors use the Mininet simulator to build numerous networks with different topologies, various nodes, and OpenFlow switches for varying SDN controllers such as beacon, POX, ONOS, IRIS, RYU, ODL, and floodlight. In [16] the author obtained results for comparing the characteristics of centralized and distributed controllers by focusing on the performance, bandwidth, and round-trip time of Floodlight and Open Daylight (ODL) controllers. The results reveal that ODL outperforms Floodlight in the growing nature of a single topology. Floodlight, on the other hand, outperforms in both linear and tree topology. Additionally, the author recommends a centralized controller over a distributed controller for complex structures. However, the author of [11] determined that Open Daylight (ODL) as a distributed controller is the best choice for a full-detailed SDN controller. As a result, the author proposed that future research compare ODL or RYU controllers with further implementations.

According to previous research, most researchers concentrated on analysing either centralized or multi-controller architectures separately. Few research has used the Mininet emulator and Ryu controller to conduct a direct and comprehensive comparison of these architectures. The motivation for this research was the need to fill this research gap. The proposed study will contribute to the existing body of knowledge on controller architectures in SDN by addressing these research gaps and providing a comprehensive evaluation and comparison of centralized and multi-controller architectures. This research study has significance for academics, researchers, network architects, and operators to acquire a better understanding of each architecture's performance features, strengths, and weaknesses. It will aid in making educated judgments regarding the selection of controller architecture for specific network requirements. Therefore, the objective of this study is to evaluate and compare centralized and multi-controller architectures in SDN using the Mininet emulator and Ryu Controller. The effects of centralized and multi-controller architectures on network jitter, throughput, and memory usage are examined.

2. Methodology

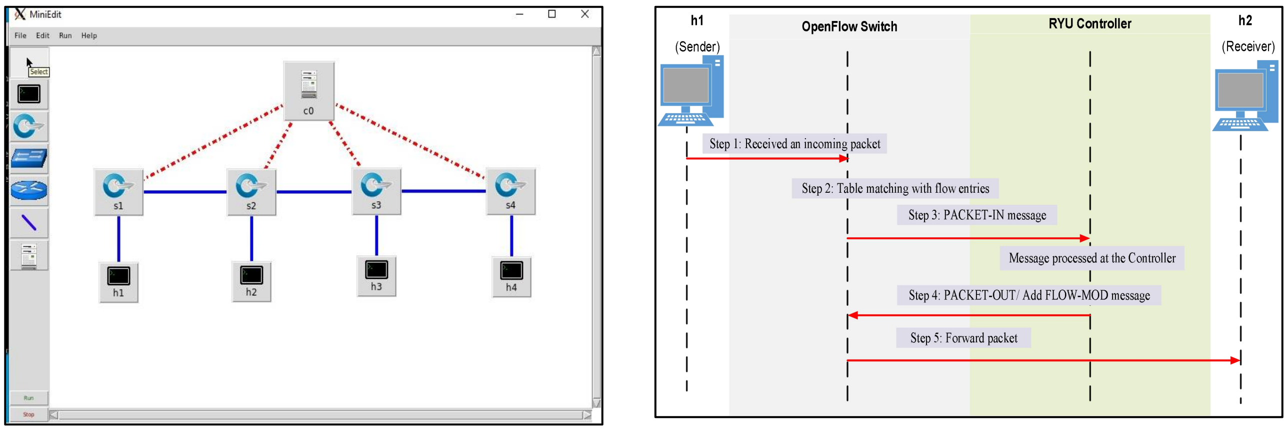
The Mininet network emulator is used to construct three networks for this study. Mininet includes MiniEdit, a simple graphical user interface editor, and CLI-based commands for designing standard or custom topologies. This study utilized two virtual machines. The first system executes Mininet, which emulates network topology, while the second system runs the Ryu controller. The two virtual machines must be interconnected and run experiment-required services, including SSH, the X Server software client, and Wireshark. Mininet Python API version 2.3.0d1 was installed together with Open vSwitch (OVS) version 2.5.4 and Ryu controller on Ubuntu 20.04.4 LTS. In addition, the Xming server was employed to visualize and generate traffic between the source and destination in Wireshark. The configurations are constructed with Open Vswitches (OVS) software switches, and the Southbound control traffic protocol is OpenFlow version 1.3.

Three custom network topologies are constructed with four switches and four hosts communicating to a single controller as a centralized topology, and two multi-controller topologies with two controllers performing equal and master-slave roles, respectively. The RYU controller has been deployed in the control plane via ports 6653 and 6654 and the loopback IP address 127.0.0.1. The range of IP addresses assigned to all hosts and switches is 10.1.1.0/24. After network topologies have been established, tools such as ping and iperf are used to compare the performance of centralized and multi-controller SDN architectures in terms of throughput under

TCP data flow, jitter, and memory utilization. The Wireshark protocol analyzer is utilized to monitor and gather network traffic statistics.

2.1 Centralized SDN Controller Architecture

In the initial stage of SDN design, a single controller manages the entire network. In Figure 3(a) exhibits the network topologies created in Mininet connected to an RYU controller (C0) and manages four switches in the network. Figure 3(b) shows the sequence processes of OpenFlow packet processing in centralised network topology. In this example, when the source host (h1) sends a new packet to the OpenFlow switch (s1), the s1 sends (packet IN) messages to the RYU controller (C0) to get the routing for the new packet. After getting the response message from the controller, the switch forwards the packet to the destination device. Finally, the packet reaches the destination host (h2) successfully. The controller plays a major role in the process of traffic transmission. Ping command tests were run inside Mininet CLI to test the connection in each topology. The data traffic transmitted from node to node is monitored using Wireshark. The performance is measured in terms of throughput under TCP data flow, and the jitter and 'top' commands are used to display real-time information on the controller's memory usage.


MiniEdit
File Edit Run Help
c0
s1
s2
s3
s4
h1
h2
h3
h4
h1
(Sender)
OpenFlow Switch
RYU Controller
h2
(Receiver)
Step 1: Received an incoming packet
Step 2: Table matching with flow entries
Step 3: PACKET-IN message
Message processed at the Controller
Step 4: PACKET-OUT/ Add FLOW-MOD message
Step 5: Forward packet

(a) (b)

Fig. 3. SDN network topology (a) Centralized network topology (b) Sequence processes of OpenFlow packet in centralized network topology

2.2 Multi-Controller SDN Architecture

According to [17], a switch could be connected to several controllers simultaneously. A switch may have a maximum of one controller in the master state and multiple controllers in equal roles, or slave states, connected at the same time. Each controller may send a role request message to the switch to inform it of its role, and the switch is required to keep track of each controller connection's role. Therefore, to simulate the performance of multi-controller SDN architecture, network topology with two controllers has been designed. Figure 4(a) shows the topology of multi-controller SDN architecture with equal roles. The controller with an equal role has complete access to the OpenFlow switch. It receives all asynchronous communications from the switch and transmits commands to change its state as illustrated in Figure 4(b). Figure 5(a) exhibits that each switch is connected to a single master controller and a slave controller. Figure 5(b) illustrates the slave role controller that has read-only access to the switch. It does not receive asynchronous messages, unlike the master role controller, which has full access to the switch, which is the same finding as earlier research by [18-19]. In addition, the master controller is used to process packet-IN requests from switches, while the slave controllers serve as backups.

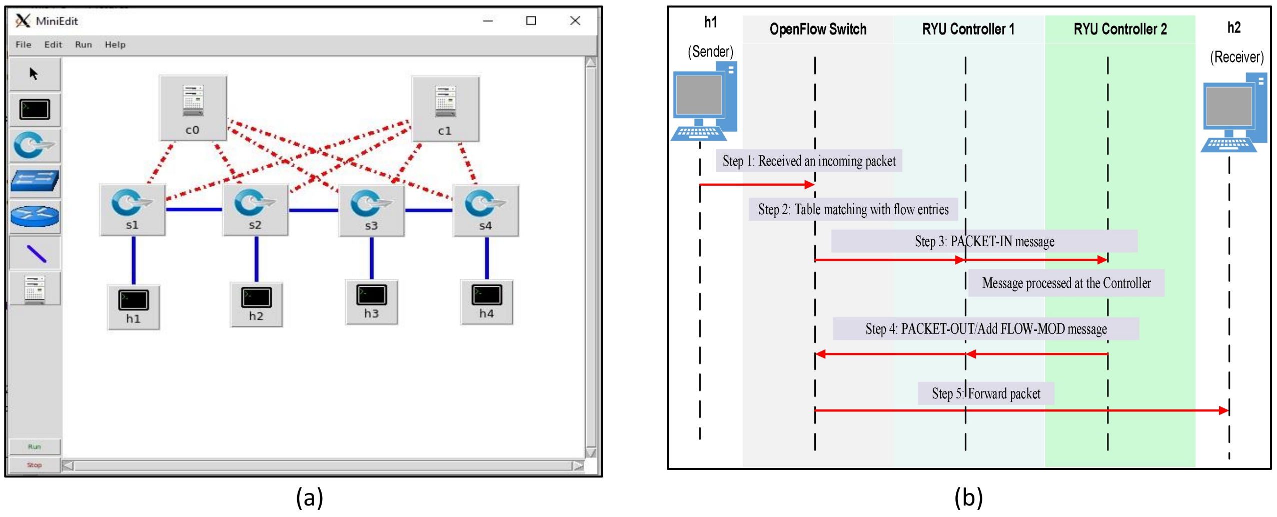

MiniEdit
File Edit Run Help
c0
c1
s1
s2
s3
s4
h1
h2
h3
h4
Run
Stop
(a)
h1
(Sender)
OpenFlow Switch
RYU Controller 1
RYU Controller 2
h2
(Receiver)
Step 1: Received an incoming packet
Step 2: Table matching with flow entries
Step 3: PACKET-IN message
Message processed at the Controller
Step 4: PACKET-OUT/Add FLOW-MOD message
Step 5: Forward packet
(b)

Fig. 4. SDN network topology (a) multi-controller with equal role network topology (b) Sequence processes of OpenFlow packet in multi-controller with equal role network topology

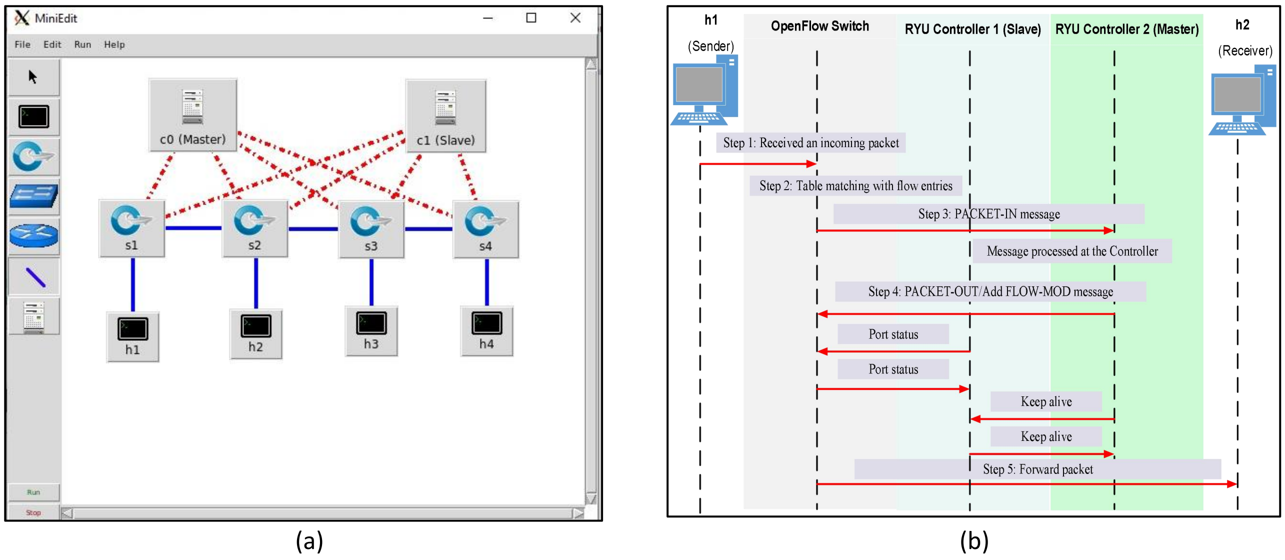


Fig. 5. SDN network topology (a) multi-controller with master-slave role network topology (b) Sequence processes of OpenFlow packet in multi-controller with master-slave role network topology

After establishing all connections effectively, network traffic is established between all nodes. Using the ping command and the iperf tool, the performance of the multi-controller network and the ability of the multi-controller to use the different roles in the SDN network were observed. Iperf is a widely used network testing application that can generate TCP and UDP data streams and measure the throughput of a network by generating traffic between two nodes in network topology as described in [15-20]. One host utilized it as a sender or client, while the other utilized it as a receiver or server. Generated TCP and UDP traffic will be allocated specific bandwidth, time interval, window size, and buffer size. Therefore, the TCP and UDP protocols are selected for evaluation in this study. The throughput performance metrics for TCP traffic are evaluated by running simulations between the two nodes for 10 minutes per simulation. In addition, jitter is considered for UDP traffic through the allocation of bandwidth between 10Mbps, 20Mbps, 30Mbps, 40Mbps, and 50Mbps. After executing the iperf traffic generator on an Ubuntu terminal, the data was gathered for analysis.

3. Results

3.1 Throughput

To determine how many packets the controller can handle in a second, throughput tests are used. The quantity of data transferred per unit of time determines the throughput of a network. After the experiment was studied, throughput network analysis graphs were made that showed the anticipated outcomes. The throughput has been evaluated using iperf command “<node> iperf -s” and “<node> iperf -c <node> -i 1 -t 10”, which was executed for 10 seconds on the client and gathered data every 1 second on the server. The result of average throughput between each node for the centralized, equal role controller and master-slave controller is shown graphically in Figure 6.

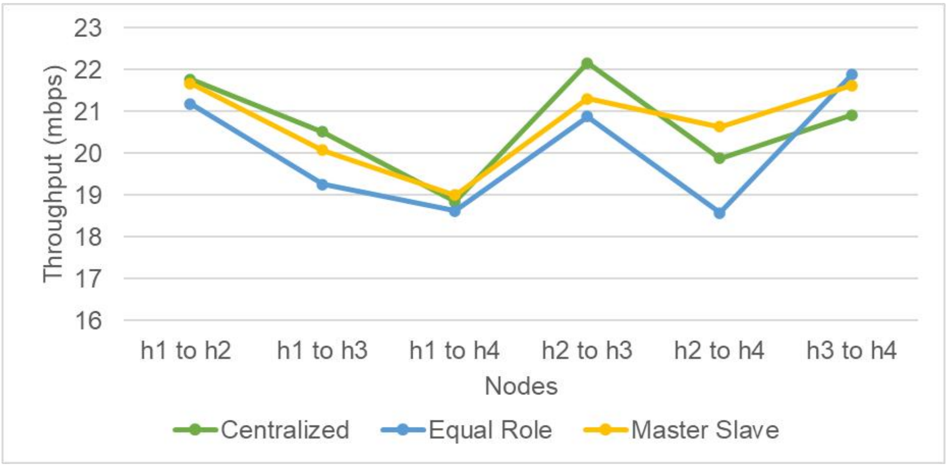


Fig. 6. Comparison of throughput for three SDN controller architecture

Based on the findings of the throughput comparison across the three SDN network controller architectures, it was discovered that the centralized SDN network transmission between h2 and h3 nodes obtained the greatest throughput of 22.16 Gigabits. In contrast, the multi-controller SDN network with equal roles showed the lowest throughput of 18.57 Gigabits between the h2 and h4 nodes. These findings illustrate the influence of various network controller architectures on throughput performance. It appears that the centralized architecture, in which a single controller manages the entire network, offers advantages in terms of attaining higher throughput on specific network paths. In contrast, the multi-controller SDN network with equal roles may encounter difficulties sustaining comparable throughput levels.

3.2 Jitter

The inconsistency or irregularity in the arrival time of packets at their destination is referred to as jitter. This test has used UDP packets where packets are sent to the receivers without any checks. To measure jitter, the iperf utility has been executed on different bandwidths under UDP node-to-node connection using iperf command “<node> iperf -u -s” and “<node> iperf -u -c <node> -b <bandwidth>”. Based on the comparison of jitter among the three SDN network controller architectures in Figure 7, the results indicate that all the networks generally exhibited low jitter levels, typically below 0.02ms. However, the multi-controller SDN network with equal roles at a bandwidth of 30Mbps displayed the highest jitter with a value of 0.06ms. The slight increase in jitter observed in the multi-controller SDN network with equal roles at higher bandwidth could be attributed to various factors, such as increased network congestion, suboptimal packet scheduling, or buffer management.

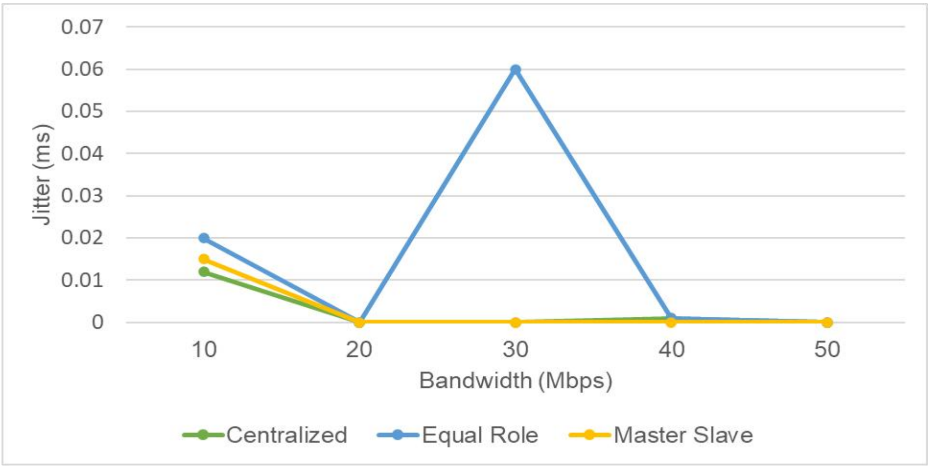


Fig. 7. Comparison of jitter for three SDN controller architecture

The analysis indicates that both centralized and multi-controller SDN architectures can guarantee the delivery of time-sensitive data with minimal jitter. The centralized architecture reveals consistently reliable performance, whereas the multi-controller architecture with equal roles performs well but may require optimization to reduce jitter in certain circumstances.

3.3 Memory Usage

It is essential to observe that memory usage and management can vary based on the controller implementation, network size, and enabled features. Memory utilization must be optimised to ensure optimal performance, responsiveness, and scalability of the SDN network controller. In this study, the 'top' command was used to observe the controller's real-time memory consumption in centralized and multi-controller SDN networks. The ‘top’ command is a Linux utility that provides a summary of the current state of the system.

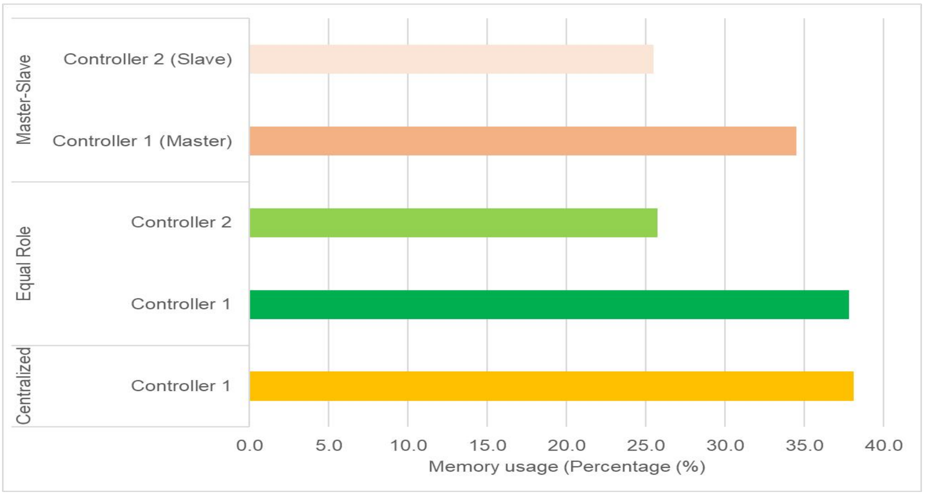


Fig. 8. Comparison of memory usage for three SDN controller architecture

The comparison of the memory usage percentage on three networks is shown in Figure 8. As can be seen, the higher memory usage is 38.09% in the centralized SDN controller network and the lowest memory usage is 25.46% in the controller slave role in multi-controller SDN (master-slave role). These results demonstrate that equal role controller 2 and the slave controller in multi-controller SDN (master-slave role) configuration have comparatively lower memory usage than the centralized controller and equal role controller 1. The lower memory usage could be due to the reduced responsibilities and functionalities assigned to the slave controller in this configuration. Meanwhile, higher memory usage can be defined by various factors, including storing network topology information, managing flow tables, processing events, and supporting various controller applications and modules.

4. Conclusions

In conclusion, this study used the Mininet emulator and Ryu controller to assess and compare the performance of centralised and distributed control systems in SDN. This study examined the effects of centralized and multi-controller SDN architectures on network throughput, jitter, and memory consumption. The study discovered differences in network throughput among the architectures. According to the result, the centralized architecture outperformed the equal role architecture in terms of achieving higher throughput. Additionally, the findings show that all the evaluated architectures displayed generally low levels of network jitter, with most values falling below 0.02ms. The multi-controller SDN network, on the other hand, showed significantly higher jitter with a value of 0.06ms at a bandwidth of 30Mbps for equal role architecture. This result indicates that the equal role configuration would need more optimization to reduce jitter in particular circumstances. The study discovered that the centralized architecture requires more memory resources than the equal role configuration in terms of memory consumption. Therefore, it can be concluded that the choice of controller architecture has a significant impact on network performance. Compared to the multi-controller equal role configuration, the centralized control architecture exhibited a higher throughput, but a higher memory consumption. Moreover, while all architectures maintained minimal levels of jitter, the equal role configuration exhibited slightly higher jitter in certain circumstances.

This study highlights the significance of appropriately constructing and optimizing SDN controller architecture. When deciding between centralized and distributed controller architectures, network administrators should evaluate the trade-offs between throughput, memory utilization, and jitter. Further research and optimization efforts can be made to improve the efficiency of the equal-role multi-controller configuration and reduce jitter. Overall, this study provides valuable insights into the performance characteristics of different control architectures in software-defined networks.

Acknowledgement
This paper is a preliminary study of the first author's ongoing research work under the main supervision of Dr Mohd Nazri Mohd Warip, Associate Professor, University Malaysia Perlis for her PhD program. This research was not funded by any grant. We also want to express our gratitude to the reviewers of this manuscript for their valuable comments throughout the process.

References

[1] Aziz, Mohd Zafran Abdul, and Koji Okamura. "A security trending review on software define network (SDN)." *Journal of Advanced Research in Computing and Applications* 6, no. 1 (2016): 1-16.

[2] Segeč, P., M. Moravčik, J. Uratmová, J. Papán, and O. Yeremenko. "SD-WAN-architecture, functions and benefits." In *2020 18th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pp. 593-599. IEEE, 2020. https://doi.org/10.1109/ICETA51985.2020.9379257

[3] Kelian, Virakwan Hai, Mohd Nazri Mohd Warip, R. Badlishah Ahmad, Phaklen Ehkan, Fazrul Faiz Zakaria, and Mohd Zaizu Ilyas. "Toward Adaptive and Scalable Topology in Distributed SDN Controller." *Journal of Advanced Research in Applied Sciences and Engineering Technology* 30, no. 1 (2023): 115-131. https://doi.org/10.37934/araset.30.1.115131

[4] Qiu, Kun, Jin Zhao, Xin Wang, Xiaoming Fu, and Stefano Secci. "Efficient recovery path computation for fast reroute in large-scale software-defined networks." *IEEE Journal on Selected Areas in Communications* 37, no. 8 (2019): 1755-1768. https://doi.org/10.1109/JSAC.2019.2927098

[5] Zafar, Samra, Zefeng Lv, Nizam Hussain Zaydi, Muhammad Ibrar, and Xiaopeng Hu. "DSMLB: Dynamic switch-migration based load balancing for software-defined IoT network." *Computer Networks* 214 (2022): 109145. https://doi.org/10.1016/j.comnet.2022.109145

[6] Priyadarsini, Madhukrishna, and Padmalochan Bera. "Software defined networking architecture, traffic management, security, and placement: A survey." *Computer Networks* 192 (2021): 108047. https://doi.org/10.1016/j.comnet.2021.108047

[7] Islam, Md Tariqul, Nazrul Islam, and Md Al Refat. "Node to node performance evaluation through RYU SDN controller." *Wireless Personal Communications* 112 (2020): 555-570. https://doi.org/10.1007/s11277-020-07060-4

[8] Virdis, Antonio, and Michael Kirsche. "Recent Advances in Network Simulation." *EAI/Springer Innovations in Communication and Computing* (2019). https://doi.org/10.1007/978-3-030-12842-5

[9] Abdullah, Mahmood Z., Nasir A. Al-Awad, and Fatima W. Hussein. "Performance Comparison and Evaluation of Different Software Defined Networks Controllers." *International Journal of Computing and Network Technology* 6, no. 2 (2018). http://dx-doi.org/10.12785/ijcnt/060201

[10] Bhardwaj, Shanu, and Surya Narayan Panda. "Performance evaluation using RYU SDN controller in software-defined networking environment." *Wireless Personal Communications* 122, no. 1 (2022): 701-723. https://doi.org/10.1007/s11277-021-08920-3

[11] Gupta, Neelam, Mashael S. Maashi, Sarvesh Tanwar, Sumit Badotra, Mohammed Aljebreen, and Salil Bharany. "A comparative study of software defined networking controllers using mininet." *Electronics* 11, no. 17 (2022): 2715. https://doi.org/10.3390/electronics11172715

[12] Bhattacharyya, Arindam, and P. Prakasam. "Performance Evaluation of Various Topological Software Defined Networks Using Mininet Simulator and Pox Controller." In *2022 Second International Conference on Next Generation Intelligent Systems (ICNGIS)*, pp. 1-5. IEEE, 2022. https://doi.org/10.1109/ICNGIS54955.2022.10079779

[13] Salam, Rakesh, and Ansuman Bhattacharya. "Performance evaluation of SDN architecture through D-ITG platform for distributed controller over single controller." In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pp. 1-6. IEEE, 2021. https://doi.org/10.1109/ICCCNT51525.2021.9579724

[14] Ahmed, Hatim Gasmelseed, and R. Ramalakshmi. "Performance analysis of centralized and distributed SDN controllers for load balancing application." In *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 758-764. IEEE, 2018. https://doi.org/10.1109/ICOEI.2018.8553946

[15] Alraawi, Abdulmaged Ali M., and Sami Abbas Nagar Adam. "Performance evaluation of controller based sdn network over non-controller based network in data center network." In *2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, pp. 1-4. IEEE, 2021. https://doi.org/10.1109/ICCCEEE49695.2021.9429642

[16] Research, Pushpa J., and Pethuru Raj. "Topology-based analysis of performance evaluation of centralized vs. distributed SDN controller." In *2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC)*, pp. 1-8. IEEE, 2018. https://doi.org/10.1109/ICCTAC.2018.8370394

[17] Pfaff, Ben, Bob Lantz, and Bea Heller. "Openflow switch specification, version 1.3. 0." *Open Networking Foundation* 42 (2012).

[18] Blial, Othmane, Mouad Ben Mamoun, and Redouane Benaini. "An overview on SDN architectures with multiple controllers." *Journal of Computer Networks and Communications* 2016, no. 1 (2016): 9396525. https://doi.org/10.1155/2016/9396525

[19] Ryait, Deepjyot Kaur, and Manmohan Sharma. "To Eliminate the Threat of a Single Point of Failure in the SDN by using the Multiple Controllers." https://doi.org/10.35940/ijrte.B3433.079220

[20] Naing, May Thae, Thiri Thitsar Khaing, and Aung Htein Maw. "Evaluation of tcp and udp traffic over software-defined networking." In *2019 International Conference on Advanced Information Technologies (ICAIT)*, pp. 7-12. IEEE, 2019. https://doi.org/10.1109/AITC.2019.8921086