OPEN ACCESS

Volume 10 (4), October 2026, 1487-1497

E-ISSN:2580-1643

Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi)

DOI: https://doi.org/10.35870/jtik.v10i4.6338

Comparative Analysis of Shortest Path and Load Balancing Algorithms in Software-Defined Networking Using Ryu Controller and Mininet

Neca Aqila [1*]

[1*] *Information System Study Programs, School of Indsutrial Engineering, Telkom University, Bandung City, West Java Province, Indonesia.*

a r t i c l e i n f o

Article history:
Received 15 January 2026
Received in revised form
3 March 2026
Accepted 25 April 2026
Available online October
2026.

Keywords:
Software-Defined Networking; Routing Algorithms; Shortest Path; Load Balancing; Ryu Controller; Mininet.

Kata Kunci:
Software-Defined Networking; Algoritma Algorithms; Shortest Path; Penyeimbangan Beban; Ryu Controller; Mininet.

a b s t r a c t

Software-Defined Networking (SDN) enables programmable network management through the separation of the control plane and the data plane, making routing algorithm selection a critical factor in traffic distribution and network performance. The shortest path algorithm is commonly used as a baseline due to its deterministic behavior, but it may lead to load imbalance in topologies with multiple available paths. This study aims to compare routing behavior and network throughput between shortest path and Round Robin–based load balancing algorithms in an SDN environment. An experimental approach was employed by implementing both algorithms on the same topology and configuration using the Ryu controller and Mininet. Evaluation was conducted using observational and descriptive analysis through routing path inspection, OpenFlow flow rule analysis, and throughput measurement using iperf, a widely used network performance measurement tool that generates traffic to evaluate bandwidth between end hosts. The experimental results show that the shortest path algorithm consistently utilizes a single dominant path with an aggregate throughput of 6.46 Gbit/s, while the load balancing algorithm distributes traffic across multiple active paths and achieves a higher aggregate throughput of 9.65 Gbit/s. These findings indicate that load balancing provides better bandwidth utilization than shortest path under the tested experimental conditions.

a b s t r a k

Software-Defined Networking (SDN) memungkinkan manajemen jaringan yang dapat diprogram melalui pemisahan bidang kontrol dan bidang data, menjadikan pemilihan algoritma routing sebagai faktor penting dalam distribusi lalu lintas dan kinerja jaringan. Algoritma jalur terpendek umumnya digunakan sebagai dasar karena perilakunya yang deterministik, tetapi dapat menyebabkan ketidakseimbangan beban dalam topologi dengan banyak jalur yang tersedia. Studi ini bertujuan untuk membandingkan perilaku routing dan throughput jaringan antara algoritma penyeimbangan beban berbasis jalur terpendek dan Round Robin dalam lingkungan SDN. Pendekatan eksperimental digunakan dengan mengimplementasikan kedua algoritma pada topologi dan konfigurasi yang sama menggunakan pengontrol Ryu dan Mininet. Evaluasi dilakukan menggunakan analisis observasional dan deskriptif melalui inspeksi jalur routing, analisis aturan aliran OpenFlow, dan pengukuran throughput menggunakan iperf, alat pengukuran kinerja jaringan yang banyak digunakan yang menghasilkan lalu lintas untuk mengevaluasi bandwidth antar host akhir. Hasil eksperimen menunjukkan bahwa algoritma jalur terpendek secara konsisten memanfaatkan satu jalur dominan dengan throughput agregat sebesar 6,46 Gbit/s, sedangkan algoritma penyeimbangan beban mendistribusikan lalu lintas ke beberapa jalur aktif dan mencapai throughput agregat yang lebih tinggi yaitu 9,65 Gbit/s. Temuan ini menunjukkan bahwa penyeimbangan beban memberikan pemanfaatan bandwidth yang lebih baik daripada jalur terpendek dalam kondisi eksperimen yang diuji.

Corresponding Author. Email: necqila@gmail.com [1].

1. Introduction

Software-Defined Networking (SDN) is a networking paradigm that enables programmable network management through the separation of the control plane and the data plane (Hamdan *et al.*, 2021; Kazi *et al.*, 2025; Tache *et al.*, 2024; Joshi and Gupta, 2024). This architecture provides high flexibility in routing decision-making because the controller has a global view of the network topology and traffic flows. This condition makes SDN a platform widely utilized in experimental research to evaluate routing algorithms and traffic management in controlled and easily reproducible environments (Hussain *et al.*, 2022). In the context of SDN, routing algorithms play an important role because they directly affect traffic distribution and network performance. The shortest path algorithm is the most commonly used routing approach and is often employed as a baseline in research due to its simplicity of implementation and deterministic nature (Abdelghany *et al.*, 2022; Kumar and Thakur, 2023). In SDN environments, the shortest path algorithm can be implemented centrally on controllers such as Ryu by utilizing network topology information obtained through OpenFlow (Patel *et al.*, 2024; Linsheng *et al.*, 2022).

However, several studies indicate that the shortest path algorithm has limitations under increasing traffic conditions. Repeated selection of the same path can lead to load imbalance and potential congestion on certain links, particularly in network topologies that provide alternative paths (El-Hefnawy *et al.*, 2021; Yusuf *et al.*, 2023). These limitations encourage the development of load balancing mechanisms in SDN that leverage the controller's global visibility to distribute traffic flows across multiple available paths. Load balancing approaches in SDN, including simple algorithms such as Round Robin, are designed to distribute traffic loads more evenly without relying solely on a single shortest path (Omer *et al.*, 2021; Singh *et al.*, 2022). Several studies show that load balancing can improve network resource utilization and aggregate throughput, even though it uses routing decision-making mechanisms that differ from traditional shortest path routing (Prabakaran and Ramar, 2021; Belgaum *et al.*, 2020). Load balancing implementations on the Ryu controller are also widely used in experiments to analyze traffic distribution and SDN network performance (Bhardwaj and Girdhar, 2023; Alssaheli *et al.*, 2023; Bhardwaj and Panda, 2022). In addition, recent studies have proposed various routing optimization algorithms and adaptive approaches in SDN, such as the use of k-shortest path and dynamic routing optimization, to address the limitations of basic algorithms (El-Hefnawy *et al.*, 2021; Yusuf *et al.*, 2023; Hamdan *et al.*, 2021). Although these approaches demonstrate potential performance improvements, higher complexity and variations in experimental scenarios often make direct comparison with basic routing algorithms more difficult. Based on the literature review, it can be identified that most previous studies discuss shortest path algorithms, load balancing, or routing optimization separately, or evaluate them in experimental environments that differ in terms of topology, controller, and system configuration (Bhardwaj and Panda, 2022; Badotra and Panda, 2020; Patil *et al.*, 2020). This condition results in limited comparative understanding of the fundamental characteristics of the two routing approaches. Experimental studies that directly compare shortest path and load balancing algorithms on the same topology and simulation environment, particularly using the Ryu controller and Mininet, remain relatively limited (Bhardwaj and Girdhar, 2023).

Therefore, this study emphasizes the importance of comparative evaluation of basic routing algorithms in a controlled and consistent SDN environment. Evaluation of basic algorithms is required as a foundation for understanding the fundamental characteristics of each approach before assessing the effectiveness of more complex routing algorithms (Tache *et al.*, 2024; Prabakaran and Ramar, 2021). In addition, the use of simple and easily reproducible simulation scenarios is relevant to improve experimental validity and result reproducibility (Patil *et al.*, 2020). Based on the background and research gap, the objective of this study is to conduct a comparative analysis between shortest path and load balancing algorithms in a Software-Defined Networking environment using the Ryu controller and Mininet. This study explicitly focuses on evaluating path selection behavior and network throughput produced by each algorithm under

identical topology and simulation configurations, thereby providing a controlled and reproducible experimental understanding of the differences in routing characteristics between shortest path and load balancing in SDN.

2. Research Methodology

Research Design

This study employs an experimental approach to analyze and compare the shortest path and load balancing routing algorithms in a Software-Defined Networking (SDN) environment. This approach is selected because the study does not aim to propose a new algorithm, but rather to evaluate the behavior of two existing routing algorithms through direct implementation and testing in a controlled SDN network. The research design is conducted by applying both algorithms to the same network topology, controller, and simulation environment to ensure a fair and consistent comparison. The experiments are executed separately for each algorithm, starting with topology initialization and connectivity testing, followed by observation of routing behavior. This approach is descriptive and comparative in nature, without involving mathematical modeling or further optimization, so that all analyses are based solely on experimental results that are actually obtained and can be replicated using the same configuration.

Software-Defined Networking Architecture

The Software-Defined Networking architecture used in this study follows the concept of separation between the control plane and the data plane, where routing decision-making is centralized at the controller, while network devices merely forward packets based on the rules provided. This separation allows the controller to have a global view of the network topology and traffic flows, enabling routing policies to be applied centrally and consistently (Hamdan *et al.*, 2021). The Ryu controller is used as the control plane because it is modular, Python-based, and widely utilized in experimental SDN research for the implementation and evaluation of routing algorithms (Bhardwaj and Girdhar, 2023). The data plane is emulated using OpenFlow switches and hosts running in the Mininet environment, where switches forward packets based on flow rules installed by the controller through the OpenFlow protocol (Patil *et al.*, 2020). The interaction between the controller and switches remains consistent across all experimental scenarios, for both the shortest path and load balancing algorithms. By using the same SDN architecture for all tests, the observed differences in routing behavior can be directly attributed to the applied algorithms rather than to changes in architecture or system configuration, ensuring that the experimental results are controlled and reproducible.

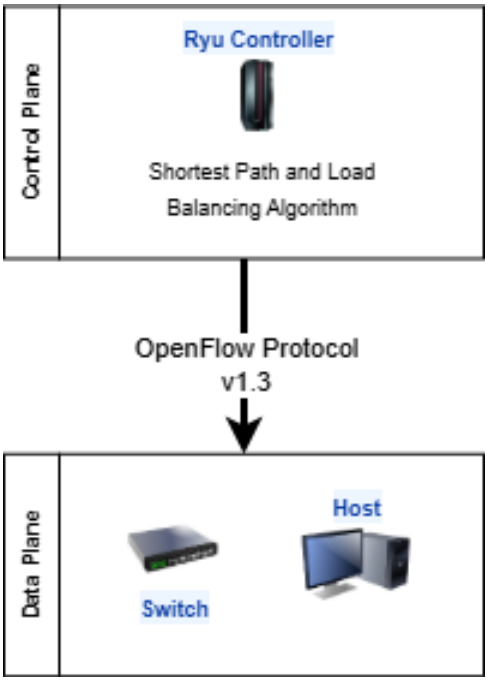

Figure 1. Software-Defined Networking Architecture

Routing Algorithm Implementation

This subsection explains the implementation of the shortest path and load balancing routing algorithms in a Software-Defined Networking (SDN) environment, focusing on the algorithmic logic executed at the SDN controller without involving further optimization. The shortest path algorithm is implemented as a basic routing mechanism that determines packet delivery paths based on the shortest distance or minimum number of hops between nodes. With global network topology information, the controller centrally computes the shortest path and installs flow rules on OpenFlow switches. This approach is widely used as a baseline routing method due to its deterministic nature and ease of implementation on controllers such as Ryu (Abdelghany *et al.*, 2022; Kumar and

Thakur, 2023; Patel *et al*., 2024). As a comparison, the load balancing algorithm is implemented using a Round Robin approach that distributes traffic flows alternately across multiple available paths, without relying on a single shortest path. This implementation is carried out without considering additional network metrics to maintain algorithmic simplicity and ensure a fair comparison with the shortest path approach. Both algorithms are tested on the same SDN architecture and network topology, so that the observed differences in routing behavior can be directly attributed to the fundamental characteristics of the applied shortest path and load balancing algorithms (Omer *et al*., 2021; Prabakaran and Ramar, 2021).

Experimental Setup

The experiments in this study are conducted using Mininet as the network emulation platform and Ryu as the Software-Defined Networking controller. Mininet is used because it is capable of emulating an SDN network within a single integrated environment consisting of hosts, OpenFlow switches, and a controller, and it supports lightweight and reproducible experiments (Patil *et al*., 2020). The Ryu controller is run externally and serves as the control plane that implements routing logic according to the evaluated algorithms. All experimental scenarios are executed in the same simulation environment with identical baseline configurations. In each scenario, only one routing algorithm is activated on the controller, either shortest path or load balancing, to ensure that no configuration conflicts occur and that the observed routing behavior fully reflects the algorithm being tested (Bhardwaj and Girdhar, 2023). No changes are made to the controller, OpenFlow version, or communication mechanisms between the controller and switches throughout the experiments. The network topology is built using Mininet and is designed to provide more than one communication path between the source host and the destination host, so that differences in routing behavior between the shortest path and load balancing algorithms can be clearly observed. The topology consists of two hosts, five OpenFlow switches, and one SDN controller. Host h1 acts as the traffic source and is connected to switch s1, while host h2 acts as the traffic destination and is connected to switch s5. The five OpenFlow switches form two alternative paths between s1 and s5, namely the path s1 → s2 → s4 → s5 and the path s1 → s3 → s5. Throughout the experiments, the topology structure is kept static, with no addition or removal of nodes and no changes in connectivity between hosts and switches.

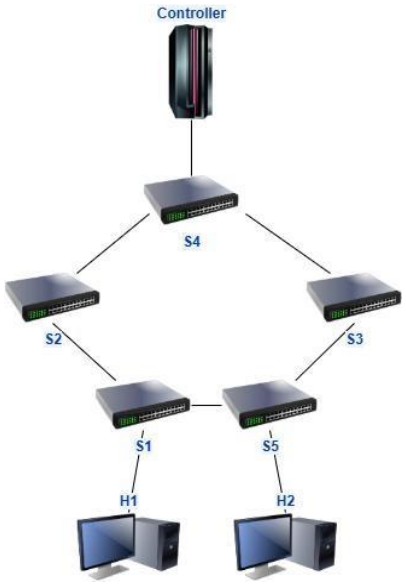

Figure 2. Network Topology Used in the Experiment

The Ryu controller is initialized using the ryu-manager command with the --observe-links option to support network topology detection. In the shortest path scenario, the controller is run by loading the ryu_sp_new.py application, while in the load balancing scenario, the controller is run with the ryu_lb.py application. Both applications use the same basic controller modules, so the differences in the resulting routing behavior originate entirely from the applied routing algorithm logic, rather than from differences in system configuration or execution environment. For each scenario, only one routing application is executed on the controller, ensuring that no flow rule conflicts occur (Bhardwaj and Girdhar, 2023).

```
mininet@mininet-vm:~$ ryu-manager --observe-links --ofp-tcp-listen-port 6633 ryu
_sp_new.py
loading app ryu_sp_new.py
loading app ryu.topology.switches
loading app ryu.controller.ofp_handler
instantiating app ryu_sp_new.py of ShortestPath
instantiating app ryu.topology.switches of Switches
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Figure 3. Execution of Ryu Controller for Shortest Path Routing

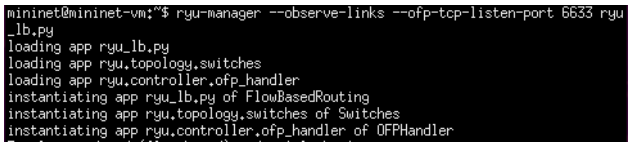

Figure 4. Execution of Ryu Controller for Load Balancing Routing

The experiments are conducted using consistently defined parameters and tools. Mininet is used to build and run the SDN network topology through Python scripts executed with Python 3, while communication between the controller and switches is carried out via the OpenFlow protocol using TCP connections (Patil *et al.*, 2020). As an initial verification step, connectivity testing is performed using the pingall command in Mininet to ensure that all hosts can communicate with each other without packet loss. All experiments are executed through a command-line interface without using external traffic generators or additional measurement tools beyond those applied in the evaluation stage, so this experimental configuration can be directly replicated in the same environment and setup (Bhardwaj and Girdhar, 2023).

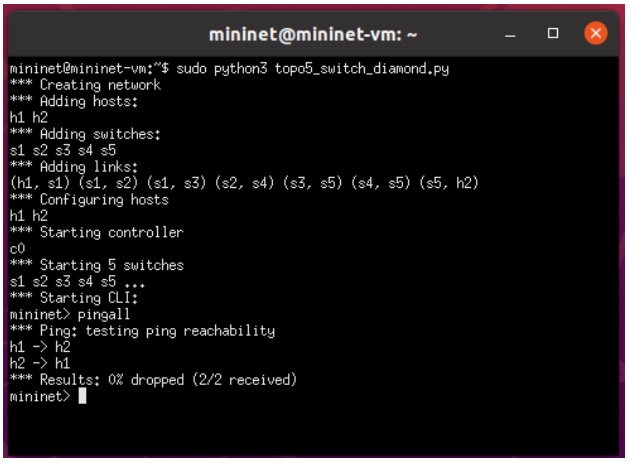

Figure 5. Mininet Topology Execution and Connectivity Verification Using Pingall

Experimental Environment Specification

The experiments were conducted using a fixed hardware and software environment within a virtual machine. The virtual machine was executed on a host system equipped with an Intel Core i5-8250U processor and 8 GB of RAM, running Ubuntu 20.04.6 LTS (64-bit) as the guest operating system. The Software-Defined Networking environment was implemented using Mininet version 2.3.0 as the network emulation platform. The SDN controller was implemented using the Ryu controller version 4.34, executed with Python 3.8.10, to manage routing decisions within the emulated network. The data plane was emulated using Open vSwitch version 2.13.8, with communication between the controller and switches based on OpenFlow version 1.3. Network throughput measurements were carried out using iperf version 2.0.13 to generate TCP traffic between source and destination hosts. The same virtualized environment and software configuration were applied consistently across all experimental scenarios.

Evaluation Procedure

The evaluation procedure in this study is designed to compare the routing behavior produced by the shortest path and load balancing algorithms within the same Software-Defined Networking environment. The evaluation is conducted in an observational and descriptive manner based on data actually obtained from the experiments, without involving measurements or metrics that are not executed. Each evaluation scenario begins with initializing the network topology in Mininet and running the Ryu controller with the appropriate routing algorithm, followed by connectivity verification between hosts to ensure that the network system functions correctly before observations are conducted. Subsequently, routing behavior is analyzed by observing the paths selected by the controller in directing traffic flows between hosts. In the shortest path scenario, observations focus on the consistency of using the shortest path as long as the topology remains unchanged, whereas in the load balancing scenario, observations focus on the variation of paths used alternately as a result of the Round Robin mechanism. The observation results from each scenario are recorded and used as the basis for comparative analysis in the results and discussion section, so that interpretations of differences in routing behavior and throughput between the two algorithms remain objective and consistent with the conducted experiments (Bhardwaj and Girdhar, 2023; Badotra and Panda, 2020).

3. Result and Discussion

Results

Shortest Path Routing Result

The experimental results for the shortest path routing scenario show that the SDN controller consistently selects a single shortest path to direct traffic from the source host (h1) to the destination host (h2). As long as the network topology does not change, the selected path remains the same for every traffic flow, reflecting the deterministic nature of the shortest path algorithm. An inspection of the flow rules installed on the OpenFlow switches is performed. The inspection is carried out using the ovs-ofctl dump-flows command on switches located along the potential paths. The results show that the flow rules direct packets to consistent output ports, without any variation toward alternative paths (Abdelghany *et al.*, 2022; Kumar and Thakur, 2023).

Figure 6. Aturan flow pada switch OpenFlow pada routing shortest path

The consistency of the flow rules installed on the switches indicates that the controller establishes a single dominant path as the packet forwarding route and maintains it throughout the experiment. No indications of flow distribution to other paths or dynamic path changes are observed. These findings confirm that the shortest path algorithm is correctly implemented and functions as the baseline routing in this study.

Load Balancing Routing Result

In the load balancing routing scenario, the experimental results show that the SDN controller does not focus traffic on a single path, but distributes traffic flows across different paths in the network topology. This behavior reflects the Round Robin–based load balancing mechanism implemented at the controller, in which multiple connection flows are directed alternately. An analysis of the flow rules installed on the OpenFlow switches is conducted to verify the paths used by each flow. The results of the ovs-ofctl dump-flows command show flow rules with different output ports, indicating that packets are forwarded through different paths in the network topology. This variation in output ports provides evidence that the controller actively performs flow distribution rather than maintaining a single dominant path (Omer *et al.*, 2021; Prabakaran and Ramar, 2021).

Figure 7. Flow pada switch OpenFlow

The presence of multiple flow rules directing traffic to different paths confirms that the load balancing mechanism operates in accordance with its intended design. In contrast to the shortest path scenario, where all traffic is concentrated on a single path, the load balancing scenario demonstrates traffic distribution across the available paths. This difference in behavior forms the basis for the comparative analysis of throughput between the two routing algorithms in the following subsection.

Throughput Comparison between Shortest Path and Load Balancing

This subsection presents a comparison of network throughput results between the shortest path routing and load balancing routing scenarios based on measurements using iperf, a widely used network performance measurement tool that generates traffic to evaluate bandwidth between end hosts. The tests are conducted with the same topology configuration and parameters, so that differences in the obtained throughput results can be directly attributed to differences in the routing mechanisms implemented at the SDN controller. In the shortest path routing scenario, the iperf test results show that all TCP connections are directed through a single dominant path. The iperf output indicates several active parallel connections, but the aggregate throughput fully depends on the capacity of the single path selected by the controller. This condition limits overall network resource utilization because the alternative paths available in the topology are not used to carry additional traffic (Singh *et al.*, 2022; Alzahrani *et al.*, 2021).

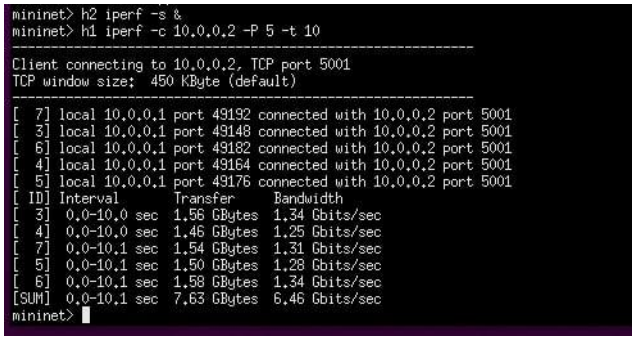

Figure 8. Iperf throughput results on shortest path routing

Conversely, in the load balancing routing scenario, the iperf test results show a significant increase in aggregate throughput. Multiple parallel TCP connections are redirected to different paths, allowing network capacity to be utilized more optimally. The iperf output shows that the total throughput achieved is higher than in the shortest path scenario, indicating that distributing traffic across more than one path has a direct impact on improving data transmission capacity (Zhang *et al.*, 2021; Kumar and Singh, 2022).

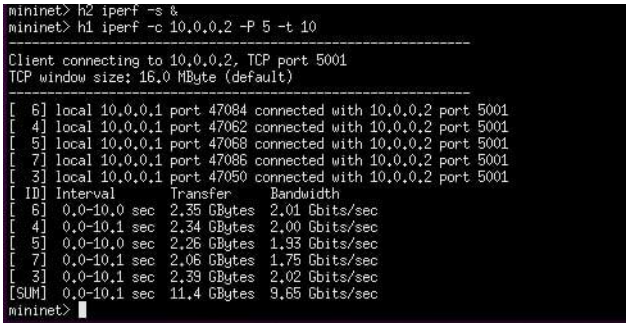

Figure 9. Iperf throughput results on routing load balancing

The differences in throughput results between the two scenarios are presented in Table 1, which shows that routing mechanisms have a direct impact on network performance, particularly in terms of bandwidth utilization. Shortest path routing tends to limit throughput because it relies on a single path, whereas load balancing routing allows multiple paths to be used simultaneously to handle parallel traffic. However, the increase in throughput in the load balancing scenario is achieved at the cost of increased complexity in flow management at the controller. Each flow must be directed and managed separately to ensure that path distribution operates according to the implemented mechanism. Therefore, the selection of a routing algorithm needs to consider network requirements, whether prioritizing simplicity and path stability or higher bandwidth utilization. These comparative results confirm that load balancing routing provides advantages in terms of throughput for the topology and testing scenarios used, while shortest path routing serves as a baseline with single-path characteristics and deterministic routing behavior (Patel *et al.*, 2022; Choudhury and Saha, 2021).

Table 1. Throughput Comparison between Shortest Path and Load Balancing Routing

Item	Qty	Unit Price	Total Amount
Makipk	1	11.48	11.48
Total	-	-	6.41

Comparative Analysis of Shortest Path and Load Balancing

To summarize the differences in routing behavior and throughput measurement results between the two evaluated routing algorithms, a comparative summary is presented in tabular form. This comparison highlights the differences in characteristics between shortest path routing, which consistently utilizes a single dominant path, and load balancing routing, which distributes traffic across multiple available paths. In addition, the table summarizes the throughput measurement results obtained using iperf, allowing the impact of each routing strategy on bandwidth utilization to be observed more concisely under the same experimental conditions.

Table 2. Comparative Analysis of Shortest Path and Load Balancing

Aspect	Shortest Path	Load Balancing
Routing Behavior	Deterministic, single path	Distributed, multiple paths
Flow Rule Pattern	Single output port	Multiple output ports
Throughput (iperf)	Lower (single-path limited)	Higher (multi-path utilization)
Path Utilization	One dominant path	Multiple active paths

Table 2 summarizes the main differences between shortest path routing and load balancing routing based on observations of routing behavior and throughput measurements. In terms of routing behavior, shortest path routing exhibits deterministic characteristics with the consistent use of a single path. This is consistent with the results of flow rule analysis, where all traffic is directed through the same path without utilizing the alternative paths available in the network topology. In contrast, load balancing routing demonstrates traffic distribution across multiple paths, as indicated by variations in flow rules and the use of more than one active path. Differences in flow rule patterns between the two algorithms further emphasize these characteristics. In shortest path routing, flow rules are directed to only one output port, reflecting the controller's selection of a single dominant route. Meanwhile, in load balancing routing, the presence of multiple output ports in the flow rules indicates that the controller actively redirects traffic flows to different paths as part of the load distribution mechanism. The impact of these differences in routing behavior is clearly reflected in the throughput measurement results obtained using iperf. Shortest path routing results in lower throughput because all traffic depends on the capacity of a single path, leading to limited bandwidth utilization.

Conversely, load balancing routing is able to achieve higher aggregate throughput by utilizing multiple paths simultaneously, thereby improving overall network resource utilization. In terms of path utilization, shortest path routing activates only one dominant path, while alternative paths remain unused throughout the experiments. In contrast, load balancing routing activates multiple paths simultaneously, resulting in more balanced traffic distribution. Overall, this table confirms that differences in path selection mechanisms between the two routing algorithms have a direct impact on traffic distribution and network throughput in the experimental scenarios used. Although the experimental results demonstrate distinct routing behavior and measurable throughput differences between the evaluated algorithms, overall SDN performance is influenced by factors beyond routing logic alone. In centralized SDN architectures, controller-related processes such as flow setup, control message exchange, and path computation introduce control-plane overhead that may affect traffic stabilization and forwarding responsiveness. In addition, routing and load balancing performance in SDN environments is often evaluated using multiple metrics, including delay, packet delivery ratio, and congestion indicators, rather than throughput alone. Variations in end-to-end latency and transient packet handling during flow installation or path alternation may therefore contribute to throughput fluctuations observed during experimentation. These parameters were not explicitly measured in the present study because the evaluation scope was deliberately limited to routing behavior and throughput under controlled experimental conditions. Future studies may extend this analysis by incorporating delay measurement, packet delivery ratio evaluation, and control-plane overhead assessment to provide a more comprehensive characterization of routing performance in Software-Defined Networking environments.

Discussion

The comparative analysis of shortest path routing and load balancing routing presented in Table 2 is consistent with the findings of previous studies in the field of Software-Defined Networking (SDN). Abdelghany *et al.* (2022) observed that shortest path routing exhibits deterministic characteristics, where a single dominant path is consistently selected for all traffic between a source and destination. This is reflected in the flow rule analysis, which shows that all traffic is directed through the same output port, without utilizing alternative paths available in the network topology. The authors noted that this deterministic behavior simplifies flow management at the controller but can lead to limited bandwidth utilization, as all traffic is constrained by the capacity of the single selected path. In contrast, the load balancing routing approach demonstrated in this study aligns with the findings of Omer *et al.* (2021) and Prabakaran and Ramar (2021). These researchers found that load balancing mechanisms in SDN can effectively distribute traffic across multiple available paths, as indicated by the variations in flow rules and the use of multiple active output ports. This distribution of traffic allows for improved overall

network resource utilization and higher aggregate throughput, as observed in the iperf measurements. The tradeoff between the simplicity of shortest path routing and the increased complexity of load balancing routing, as highlighted in this analysis, is also consistent with the discussions in the work of Singh *et al.* (2022) and Alzahrani *et al.* (2021). These authors noted that the selection of a routing algorithm for SDN environments should consider the balance between factors such as path stability, flow management overhead, and throughput performance, based on the specific requirements of the network. Overall, the comparative analysis presented in this study provides further empirical evidence supporting the conclusions drawn in previous research on the impact of routing strategies on SDN network performance and the need to carefully consider the tradeoffs involved in the selection of appropriate routing algorithms.

4. Conclusion

This study focuses on analyzing differences in routing behavior and their impact on network throughput in a Software-Defined Networking (SDN) environment through a comparison of shortest path and Round Robin–based load balancing algorithms. Through simulations conducted using Mininet and the Ryu controller, this study provides empirical insights into how routing decisions implemented at the SDN controller affect traffic distribution and network resource utilization within the same topology. The experimental results show that shortest path routing has deterministic characteristics with the consistent selection of a single dominant path. This approach offers simplicity in routing management and path stability but limits the utilization of alternative paths available in the network. In contrast, load balancing routing demonstrates the ability to distribute traffic flows across multiple paths alternately, which directly results in increased aggregate throughput based on measurements obtained using iperf. These differences confirm that path selection strategies at the SDN controller not only influence routing patterns but also contribute to the level of network bandwidth utilization. Nevertheless, the findings of this study should be interpreted within the context of existing limitations. The experiments were conducted on a relatively simple network topology, so the obtained results may not fully represent network conditions with larger scale and higher complexity. In addition, network performance evaluation focuses solely on throughput metrics, without considering other parameters such as delay, jitter, packet loss, or controller processing overhead. The applied load balancing mechanism is also static in nature and does not yet take dynamic network conditions into account. For future development, this study can be extended by applying more complex network topologies and more diverse traffic scenarios to more accurately represent real network conditions. The inclusion of more comprehensive network performance metrics and the exploration of routing or load balancing algorithms that adapt to network conditions are expected to provide deeper insights into the implications of routing algorithm selection in SDN environments.

5. References

Abdelghany, H. M., Zaki, F. W., & Ashour, M. M. (2022). Modified Dijkstra shortest path algorithm for SD networks. Original Scientific Paper.

Alssaheli, O. M. A., Zainal Abidin, Z., Zakaria, N. A., & Abal Abas, Z. Software defined network based load balancing for network performance evaluation.

Badotra, S., & Panda, S. N. (2020). Experimental comparison and evaluation of various OpenFlow software-defined networking controllers. International Journal of Applied Science and Engineering, 17(4), 317–324. https://doi.org/10.6703/IJASE.202012_17(4).317.

Belgaum, M. R., Musa, S., Alam, M. M., & Su'Ud, M. M. (2020). A systematic review of load balancing techniques in software-defined networking. IEEE Access, 8, 98612–98636. https://doi.org/10.1109/ACCESS.2020.2995849.

Bhardwaj, S., & Girdhar, A. (2023). Network traffic analysis in Software-Defined Networking using RYU controller. Wireless Personal Communications, 132(3), 1797–1818. https://doi.org/10.1007/s11277-023-10680-1.

Bhardwaj, S., & Panda, S. N. (2022). Performance evaluation using RYU SDN controller in Software-Defined Networking environment. Wireless Personal Communications, 122(1), 701–723. https://doi.org/10.1007/s11277-021-08920-3.

El-Hefnawy, N. A., Raouf, O. A., & Askr, H. (2021). Dynamic routing optimization algorithm for software defined networking. Computers, Materials and Continua, 70(1), 1349–1362. https://doi.org/10.32604/cmc.2022.017787.

Hamdan, M., Alshahrani, M., Alhassan, A., & Alshahrani, A. (2021). A comprehensive survey of load balancing techniques in software-defined network. Academic Press. https://doi.org/10.1016/j.jnca.2020.102856.

Hussain, M., Shah, N., Amin, R., Alshamrani, S. S., Alotaibi, A., & Raza, S. M. (2022). Software-defined networking: Categories, analysis, and future directions. Sensors, 22(15). https://doi.org/10.3390/s22155551.

Joshi, N., & Gupta, D. (2024). Application layer load balancing in Software Defined Networking using priority based round robin scheduling algorithm. Wireless Personal Communications, 136(2), 759–772. https://doi.org/10.1007/s11277-024-11273-2.

Kazi, B. U., Islam, M. K., Siddiqui, M. M. H., & Jaseemuddin, M. (2025). A survey on Software Defined Network-enabled edge cloud networks: Challenges and future research directions. Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/network5020016.

Kumar, D., & Thakur, J. (2023). Performance analysis of various shortest-path routing algorithms using RYU controller in SDN.

Linsheng, R., Derahman, M. N., Kadir, M. F. A., Mohamed, M. A., & Kamarudin, S. (2022). The performance effect due to varying network topologies on a software-defined network employing the k-shortest path. International Journal of Advanced and Applied Sciences, 9(6), 134–144. https://doi.org/10.21833/ijaas.2022.06.018.

Omer, Y. A. H., Mustafa, A. B. A., & Abdalla, A. G. (2021). Performance analysis of round robin load balancing in SDN. In Proceedings of the 2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE 2020). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ICCCEEE49695.2021.9429662.

Patel, K. P., Chaudhari, J. P., Mewada, H. K., Jayswal, H. S., Patel, R. V., & Kirange, D. K. (2024). Shortest path forwarding in software-defined networks using RYU controller. SSRG International Journal of Electrical and Electronics Engineering, 11(5), 299–305. https://doi.org/10.14445/23488379/IJEEE-V11I5P127.

Patil, P. B., Bhagat, K. S., Kirange, D. K., & Patil, S. D. (2020). Software defined networks using Mininet. International Journal of Recent Technology and Engineering (IJRTE), 9(1), 843–849. https://doi.org/10.35940/ijrte.F9375.059120.

Prabakaran, S., & Ramar, R. (2021). Software defined network: Load balancing algorithm design and analysis. International Arab Journal of Information Technology, 18(3), 312–318. https://doi.org/10.34028/iajit/18/3/7.

Singh, I. T., Singh, T. R., & Sinam, T. (2022). Server load balancing with round robin technique in SDN. In 2022 International Conference on Decision Aid Sciences and Applications (DASA

2022). Institute of Electrical and Electronics Engineers Inc., 503–505. https://doi.org/10.1109/DASA54658.2022.9765287.

Tache, M. D., Păscuţoiu, O., & Borcoci, E. (2024). Optimization algorithms in SDN: Routing, load balancing, and delay optimization. Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/app14145967.

Yusuf, M. N., Bakar, K. bin A. B., Isyaku, B., Osman, A. H., Nasser, M., & Elhaj, F. A. (2023). Adaptive path selection algorithm with flow classification for software-defined networks. Mathematics, 11(6). https://doi.org/10.3390/math11061404.