Electrician : Jurnal Rekayasa dan Teknologi Elektro
Vol. 20 No. 02, e-ISSN:2549-3442 http://dx.doi.org/10.23960/elc.v20.n2.3095

Evaluating the Influence of MTU Configuration on Network Performance within a Software-Defined Networking Architecture Employing Mininet and the Ryu Controller

Hane Yorda Dinata[1*], Daryan Pratama Alifi[2], Abdull Kheqal[3]

[1,2,3] Telecommunication System Universitas Pendidikan Indonesia; Dr Setia Budi Bandung Street West Java

Received: 01-02-2026
Accepted: 22-04-2026

Keywords:
Software-Defined;
Networking;
MTU;
Ryu controller;
Mininet;

Corespondent Email:
haneyorda21@upi.edu

Abstract. *This study examines how variations in Maximum Transmission Unit (MTU) affect network performance in a Software-Defined Networking (SDN) architecture using the Ryu controller, Mininet, and Open vSwitch. Experiments were conducted in a virtualized Ubuntu 18.04 environment with four MTU settings 500, 1000, 1500, and 2000 bytes and performance was evaluated through repeated measurements of Round Trip Time (RTT), throughput, jitter, and packet loss. All scenarios produced 0% packet loss, enabling focused analysis on latency, efficiency, and temporal stability. The results indicate that MTU size has a non-linear influence on SDN behavior. The 1000-byte MTU yielded the lowest RTT (10.9 ms), while larger and smaller values introduced higher delay. Throughput peaked at 1500 bytes (11.4 Mbps) but declined sharply at 2000 bytes, reflecting inefficiencies in processing oversized packets. Jitter showed a distinct pattern, remaining low at 500 bytes, increasing at mid-range MTUs, and decreasing again at 2000 bytes, suggesting sensitivity to internal buffering and queue dynamics. Overall, MTU values between 1000 and 1500 bytes offer the most balanced performance across latency, throughput, and jitter. These findings highlight the need for careful MTU selection to optimize the operational stability and efficiency of SDN-based networks.*

1. INTRODUCTION

Software-Defined Networking (SDN) has evolved from device-centric network management toward a logically centralized and programmable control paradigm that separates control-plane intelligence from data-plane forwarding functions [1]. This architectural transformation enables improved flexibility, scalability, and policy management, while simultaneously introducing new performance challenges related to controller design and system integration [2]. Empirical studies have demonstrated that SDN architectures exhibit distinctive trade-offs in throughput, latency, and responsiveness that are not readily inferred from functional specifications alone [3].

The integration of security-oriented mechanisms, including blockchain-assisted frameworks and cryptographic coordination, further complicates SDN performance characteristics by increasing processing overhead and signaling complexity [4]. Comparative evaluations of major SDN controllers indicate substantial variations in resource utilization and flow-processing efficiency, although many of these studies remain limited to emulated environments [5]. Consequently, the operational behavior of SDN systems under diverse configuration parameters remains insufficiently characterized. From a forwarding perspective, packet-level attributes such as Maximum Transmission Unit (MTU) play a critical role in shaping end-to-end performance. Foundational research on end-to-end delay guarantees highlights the importance of configuration-level parameters in achieving predictable latency behavior [15]. Similarly, early analyses of OpenFlow-based switching reveal persistent limitations in software-based forwarding related to packet handling and flow

scalability [20]. These findings indicate that configuration tuning remains an essential aspect of SDN optimization. Despite extensive research on controller architectures and forwarding mechanisms, limited attention has been devoted to isolating the influence of MTU variation within SDN environments, particularly in Python-based controllers such as Ryu. Existing studies either analyze MTU in conventional networking contexts or evaluate SDN performance without treating MTU as a primary experimental variable. As a result, the empirical relationship between MTU configuration and multi-dimensional performance metrics remains inadequately understood. Although research on the performance optimization of Software-Defined Networking (SDN) has continued to grow, a clear research gap remains in the explicit evaluation of Maximum Transmission Unit (MTU) as a primary experimental variable within SDN environments. Most previous studies have mainly focused on controller architectures, security mechanisms, or routing strategies, while MTU has generally been treated as a supporting parameter rather than the central object of analysis.

In addition, studies addressing MTU have mostly been conducted in conventional or hardware-based networks, where increasing packet size is often associated with improved transmission efficiency and throughput under certain conditions. SDN, however, exhibits different characteristics due to its additional abstraction layer, controller–switch communication, and software-based packet processing, all of which may alter the impact of MTU on network performance. Accordingly, the results of this study reveal a non-linear pattern, in which MTU influences not only throughput but also RTT and jitter differently across configurations. This finding suggests that the effect of MTU in SDN cannot simply be assumed to follow the same behavior observed in conventional networks, thereby requiring a more specific and empirical investigation. The novelty of this study lies in treating MTU variation as the main experimental variable and in providing a multi-metric evaluation that directly compares its effect on latency, throughput, jitter, and packet loss in a Ryu- and Mininet-based SDN environment.

This study addresses this gap by systematically investigating the impact of MTU settings on latency, throughput, jitter, and packet loss in a Ryu-controlled SDN environment implemented using Mininet and Open vSwitch. The objective is to establish empirical evidence that supports optimal MTU configuration and enhances performance predictability in software-defined networks. The experimental topology employed in this study consists of a single controller, one Open vSwitch instance, and two end hosts. While such a configuration may appear simplified compared to production-scale SDN deployments, it is intentionally adopted to establish a controlled and analytically tractable environment. The primary objective of this work is not to emulate large-scale network behavior, but rather to isolate and examine the direct impact of MTU variation on fundamental performance metrics without interference from confounding variables such as inter-switch routing, control-plane distribution, or heterogeneous traffic interactions. By limiting the topology to its minimal functional components, the study ensures that observed performance variations can be attributed predominantly to MTU configuration rather than to architectural complexity. This approach is consistent with experimental practices in parameter-focused SDN evaluation.

Nevertheless, real-world SDN deployments typically involve multi-switch architectures and concurrent traffic flows. These aspects are not captured in the present setup; therefore, the findings should be interpreted as baseline insights. Future work will extend the experimental design to more complex topologies to evaluate scalability and consistency under realistic conditions

2. LITERATURE REVIEW

Previous studies have emphasized the importance of MTU configuration in determining network efficiency, CPU utilization, and operational stability. Demonstrated that MTU variation significantly affects bonding-mode performance and throughput scalability [6]. In wireless and sensor-based networks, interactions between packet fragmentation and energy consumption have also been reported [7]. Measurement-oriented research has introduced MTU-aware

bandwidth estimation techniques that improve accuracy in high-speed networks, although these approaches primarily focus on measurement precision rather than achievable performance [8]. From a security perspective, fragmentation and reassembly processes have been identified as potential sources of implementation vulnerabilities [9], reinforcing the need for careful MTU management. In radio access and hardware-assisted processing environments, empirical evaluations suggest that MTU tuning can yield measurable latency reductions, albeit in a platform-dependent manner [10]. Advanced latency-aware transmission frameworks further highlight the importance of configuration-level optimization in IoT and SDN-integrated systems [11]. Similarly, SDN-based security orchestration mechanisms demonstrate the ability to mitigate RTT degradation under adversarial conditions [12]. Mobility-aware flow-caching strategies and traffic-analysis frameworks increasingly rely on RTT and throughput as indicators of control-plane responsiveness, although methodological inconsistencies persist across studies [13], [14]. Research on end-to-end delay synthesis and adaptive routing indicates growing interest in integrating optimization techniques and machine learning into SDN architectures [16]. However, these approaches are often evaluated under controlled conditions that limit their generalizability. Bandwidth characterization studies reveal structural weaknesses in centralized measurement mechanisms, motivating the adoption of distributed monitoring solutions [17]. Verification frameworks demonstrate that high forwarding integrity can be achieved with limited throughput impact, although trade-offs remain unavoidable [18]. Programmable data-plane research further underscores the strategic importance of flexible packet processing, while empirical validation remains limited [19].Recent investigations into MTU behavior in overlay networks, jumbo-frame deployments, and telemetry pipelines provide additional evidence that MTU boundaries substantially influence encapsulation overhead and performance stability [21]–[23]. Studies on research and production networks also acknowledge the relevance of packet-level parameters, although MTU effects are rarely examined in security-sensitive contexts [24].

Methodological comparisons between simulation and emulation platforms highlight the trade-offs between reproducibility and operational realism [25]. Meanwhile, advances in anomaly detection, traffic generation, and data collection tools continue to shape experimental SDN research [26], [27]. Machine learning-based traffic classification and hybrid optimization frameworks further contribute to performance enhancement, yet their integration into closed-loop QoS enforcement remains limited [28], [29]. Quality of Service (QoS) evaluation is commonly conducted using key performance indicators such as throughput, jitter, and packet loss, which collectively reflect network efficiency and transmission stability. Syanofri and Dalimunthe demonstrated that wired networks achieve higher throughput and lower jitter compared to wireless networks, while maintaining near-zero packet loss under controlled conditions, indicating superior reliability and consistency in data delivery . Their findings highlight that throughput represents transmission capacity, jitter reflects temporal stability, and packet loss indicates congestion and processing limitations. These parameters serve as fundamental benchmarks for assessing network performance and remain highly relevant in analyzing configuration-driven environments, including software-defined networks [30]. Collectively, these studies establish a foundation for analyzing MTU effects within contemporary SDN environments.

3. RESEARCH METODE

The research begins with the Start phase, followed by two parallel initial activities: Observation and Study of Literature. Observation identifies the real problem context and performance issues related to MTU variations in SDN environments, while the literature review establishes theoretical foundations, summarizes prior research, and supports methodological selection. Next, the study proceeds to Data Collection, where experimental measurements are obtained using Mininet, Ryu, and Open vSwitch with varying MTU configurations. The collected results are then prepared and examined during the Data Processing stage to ensure accuracy, consistency, and suitability for analysis. The processed findings are interpreted and

evaluated in the Results and Discussion section, presenting performance comparisons and key insights regarding the impact of MTU size on network latency and throughput. In addition to average values, statistical analysis was incorporated to quantify the variability of the observed performance metrics. For each MTU configuration, standard deviation and variance were calculated based on repeated measurements. Furthermore, 95% confidence intervals were derived to evaluate the reliability of the estimated mean values.These statistical indicators provide a more comprehensive understanding of performance stability and strengthen the validity of the experimental findings beyond descriptive analysis.

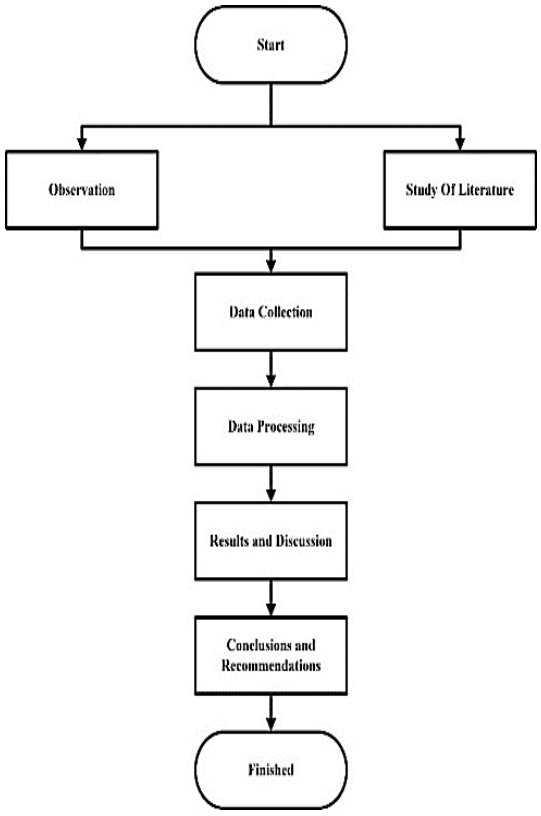

Figure 1. Process flow of the experimental study

Finally, the research produces Conclusions and Recommendations, summarizing the major outcomes and proposing optimal configurations and future improvements, before reaching the Finished stage**.** This research employs an experimental method to analyze the impact of variations in Maximum Transmission Unit (MTU) on network performance within a Software-Defined Networking (SDN) architecture. The experimental environment was executed on hardware equipped with 32 GB of RAM, an Intel Core I7 gen 11 processor, and an NVIDIA RTX 4060 GPU to ensure stable computational performance throughout the simulation. All testing was implemented using VirtualBox running the Ubuntu 18.04 operating system as a virtual machine platform where Mininet and the Ryu controller were configured. The network topology utilized consisted of one Ryu controller, one Open vSwitch switch, and two hosts. Although the experimental topology consists of one controller, one Open vSwitch switch, and two hosts, this configuration was intentionally selected as a controlled baseline to isolate the effect of MTU variation on SDN performance. A minimal topology reduces confounding factors such as multi-switch forwarding complexity, controller synchronization, and heterogeneous traffic paths. Consequently, variations in RTT, throughput, and jitter can be more directly attributed to MTU configuration rather than architectural complexity.

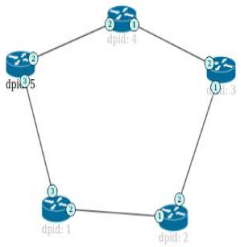

Figure 2. Network topology visualization.

The objective of this study is not to replicate the full complexity of real-world SDN deployments, but to establish a reproducible and internally valid experimental baseline. More complex topologies are considered as future work to improve external validity. In each experimental scenario, the MTU value was varied to 500, 1000, 1500, and 2000 bytes. Network performance measurements were conducted using the ping command to obtain average Round Trip Time (RTT) values and iperf3 to measure throughput. The evaluation in this study is based on ICMP and TCP traffic patterns, which represent controlled and relatively homogeneous network conditions.

While these approaches are widely used for baseline performance assessment, they do not fully capture the complexity of real-world SDN traffic, which may include concurrent flows, burst transmissions, and application-specific behaviors. The use of these tools is therefore intended to provide a controlled baseline for isolating the effect of MTU variation. Future work should incorporate more diverse traffic scenarios to better reflect operational environments and to further validate the generalizability of the findings. Each scenario was tested repeatedly to ensure result consistency, with verification that no packet loss occurred during the experiments. To ensure the reliability and reproducibility of the experimental results, each MTU configuration scenario was executed over multiple independent trials. Specifically, every scenario was repeated 10 times under identical conditions. The reported values for RTT, throughput, and jitter represent the average of these repeated measurements. Between each experimental run, the network state was reset to eliminate residual effects from previous executions, including buffer states and flow table entries. This procedure was applied to maintain consistency and to prevent temporal bias in the measurements.

All results were subsequently recorded and visualized in the form of tables and graphs to analyze the relationship between MTU variation and network performance in the SDN architecture using the Ryu controller. While the experimental design in this study was carefully structured to provide a controlled and systematic evaluation of MTU variation, several methodological limitations should be acknowledged. The network topology employed, consisting of a single controller, one switch, and two hosts, represents a relatively simplified SDN environment compared to real-world deployments. This design choice facilitates clearer isolation of MTU effects and improves measurement consistency; however, it inevitably limits the ability to capture more complex network dynamics, such as inter-switch interactions, multi-controller coordination, and heterogeneous traffic patterns. Furthermore, the use of an emulation platform based on mininet introduces inherent constraints in accurately reflecting hardware-level performance characteristics, particularly in terms of processing latency and data-plane forwarding efficiency. Consequently, the findings of this study should be interpreted within the scope of a controlled experimental setting rather than as a direct representation of operational network performance. To enhance the generalizability of the results, future research should consider more complex topologies, incorporate diverse and dynamic traffic scenarios, and extend the evaluation to different controllers as well as hardware-based network infrastructures.

4. RESULTS AND DISCUSSION

Before conducting an in-depth analysis, the measurement results were first examined to identify emerging trends regarding the impact of Maximum Transmission Unit (MTU) variation on network performance within the SDN architecture. Four MTU configurations 500, 1000, 1500, and 2000 bytes, were evaluated with respect to three primary performance indicators: Round Trip Time (RTT), packet loss, and throughput. All scenarios consistently produced 0% packet loss, allowing the performance assessment to focus primarily on latency, throughput, and jitter as the key determinants of transmission stability. The quantitative results of all experiments are summarized in Table 1 to enable systematic comparison across MTU configurations.

Table 1. The results of all the experiments

MTU	RTT Avg (ms)	Packet Loss (%)	Throughput (Mbps)	Jitter (ms)
500	44.1	0	5,41	1.7
1000	10.9	0	8,84	6
1500	23.9	0	11,4	9
2000	45.3	0	2,45	1

Table 2. Statistical dispersion of repeated RTT measurements

MTU	Std Dev	Variance	CI (95%)
500	0.116	0.0135	0.046 – 0.148 ms
1000	0.111	0.0123	0.071 – 0.169 ms
1500	0.112	0.0125	0.049 – 0.129 ms
2000	0.115	0.0138	0.052 – 0.140 ms

Table 2 summarizes the statistical dispersion and measurement uncertainty associated with the repeated RTT observations across all MTU configurations. The relatively low standard deviation values, ranging from 0.111 to 0.116 ms, indicate that the measurements are highly consistent across repeated trials. Likewise, the variance values remain small, confirming that fluctuations around the observed mean are minimal. The 95% confidence intervals are also narrow, suggesting that the dispersion estimates are stable and the experimental procedure is reproducible. Notably, the 2000-byte MTU configuration exhibits the smallest standard deviation, which indicates that its performance is consistent across runs; however, this consistency occurs alongside substantially poorer RTT and throughput performance as reported in Table 1. In other words, the 2000-byte configuration is stable but systematically inefficient.

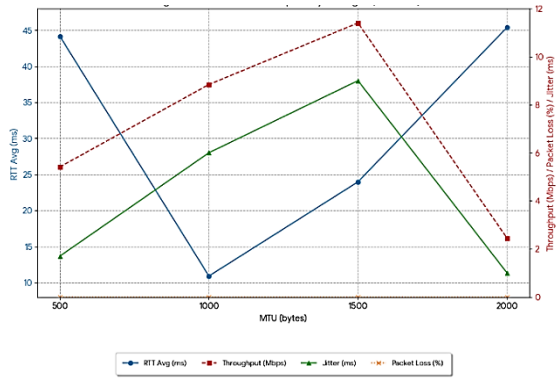

Figure 3. Visualization results of all the experiments

The results in Table 1 show that the 1000-byte MTU achieves the lowest RTT, while the 500-byte and 2000-byte configurations produce considerably higher latency. This pattern indicates that overly small MTU values increase protocol overhead, whereas excessively large MTU values impose additional processing burden within the SDN data plane. Although Table 2 shows low statistical dispersion across all configurations, such stability should be interpreted together with the QoS outcomes in Table 1. In this regard, the intermediate MTU settings, particularly 1000 and 1500 bytes, provide the most balanced performance in terms of latency, throughput, and jitter. By contrast, the 2000-byte MTU remains statistically consistent but operationally inferior, confirming that stable measurements do not necessarily correspond to efficient network behavior.

From a throughput perspective, the 1500-byte MTU delivered the highest performance at 11.4 Mbps, followed by the 1000-byte configuration. In contrast, throughput dropped sharply to 2.45 Mbps at an MTU of 2000 bytes, suggesting that oversized packets are not efficiently processed by the Open vSwitch Ryu forwarding pipeline and may contribute to internal buffering or fragmentation inefficiencies. The pronounced performance degradation observed at the 2000-byte MTU configuration warrants closer examination. One plausible explanation lies in the increased processing overhead associated with handling larger packets within a software-based forwarding environment. In the context of Mininet and Open vSwitch, packet processing is performed in software, where larger frame sizes may impose additional computational demands, particularly in terms of packet handling, queue management, and forwarding operations. In addition, the observed behavior may be related to fragmentation effects. When the effective MTU along the forwarding path is lower than the configured packet size, packets may undergo IP fragmentation, resulting in additional header overhead and requiring multiple fragments to be processed for a single transmission. Such conditions can reduce transmission efficiency and, in turn, negatively impact throughput.

Furthermore, it is reasonable to consider the role of buffering within Open vSwitch. Larger packets may occupy buffer space for longer durations, potentially leading to queue buildup and increased waiting times within the forwarding pipeline. Under constrained buffer conditions, this may contribute to transient congestion effects that further degrade performance. Taken together, these factors suggest that the sharp decline in throughput at the 2000-byte MTU is unlikely to be attributed to a single cause, but rather reflects the combined influence of fragmentation overhead, buffering dynamics, and the inherent limitations of software-based packet processing in an emulated SDN environment.

Jitter measurements provide additional insight into temporal stability. Jitter remained relatively low at 500 bytes (1.7 ms), increased at 1000 bytes (6 ms), reached its highest value at 1500 bytes (9 ms), and decreased again at 2000 bytes (1 ms). These fluctuations imply that jitter is strongly influenced by buffering and queue management within the switch, rather than being directly tied to either latency or throughput performance.

Overall, the combined observations across RTT, throughput, and jitter indicate that medium-range MTU configurations particularly between 1000 and 1500 bytes offer the most balanced performance, providing a favorable compromise between low latency, high transmission efficiency, and stable packet-timing behavior in the evaluated SDN environment.

5. CONCLUSSION

This study demonstrates that MTU configuration exerts a non-linear influence on SDN performance within a Mininet- and Ryu-based environment. Both extremely small and excessively large MTU values introduce inefficiencies, arising from increased protocol overhead and elevated packet-processing burden, respectively. In contrast, intermediate MTU sizes, particularly 1000 and 1500 bytes, provide the most favorable operational balance, with the former minimizing RTT and the latter achieving the highest throughput. Statistical dispersion analysis further confirms that the experimental results are stable and reproducible across repeated trials. Despite these findings, the use of a minimal topology and homogeneous traffic patterns limits external validity. Future work should therefore extend the evaluation to multi-switch environments, incorporate concurrent and heterogeneous traffic scenarios, and explore alternative SDN controllers to enhance the generalizability of the results.

REFERENCE

[1] A. Shirmarz and A. Ghaffari, "Performance issues and solutions in SDN-based data center: a survey," J Supercomput, vol. 76, no. 10, pp. 7545–7593, Oct. 2020, doi: 10.1007/s11227-020-03180-7.

[2] S. A. Shah, J. Faiz, M. Farooq, A. Shafi, and S. A. Mehdi, "An architectural evaluation of SDN controllers," in 2013 IEEE International Conference on Communications (ICC), Jun. 2013, pp. 3504–3508. doi: 10.1109/ICC.2013.6655093.

[3] A. Gelberger, N. Yemini, and R. Giladi, "Performance Analysis of Software-Defined Networking (SDN)," in 2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, Aug. 2013, pp. 389–393. doi: 10.1109/MASCOTS.2013.58.

[4] N. Indrason, W. Khongbuh, K. Baital, and G. Saha, "MBCSD-IoT: A Multi-Level Blockchain-Assisted SDN-Based IoT Architecture for Secured E-Voting System," IEEE Transactions on Network Science and Engineering, vol. 12, no. 3, pp. 1613–1622, May 2025, doi: 10.1109/TNSE.2025.3535726.

[5] D. Hui, S. H. Wei, L. O. Bi, X. Zhu, and S.-K. Yoon, "Performance Evaluation of Ryu, OpenDayLight and Floodlight Controllers in Diverse Software-Defined Networking Topologies," Journal of Advanced Research Design, vol. 132, no. 1, pp. 103–114, May 2025, doi: 10.37934/ard.132.1.103114.

[6] T. Demirdelen and S. Kırmızı, "Investigation of Performance Impact of 802.3ad and Round-Robin Bonding Modes Under Different MTU Configurations," Çukurova Üniversitesi Mühendislik Fakültesi Dergisi, vol. 40, no. 2, pp. 473–484, Jul. 2025, doi: 10.21605/cukurovaumfd.1664553.

[7] H. Zhao, K. Wen, S. Zhao, and S. Zhao, "A dynamic clustering routing algorithm for large-scale power wireless sensor networks based on UWB," IEEE Internet of Things Journal, pp. 1–1, 2025, doi: 10.1109/JIOT.2025.3617882.

[8] J. Huang, N. Ito, T. Oshiba, K. Satoda, and T. Murase, "PathKatana: Accurate Available Bandwidth Estimation for High-Speed Networks Using Packet Train Consisting Only of Large-Sized Packets," IEEE Open Journal of the Communications Society, vol. 6, pp. 1838–1846, 2025, doi: 10.1109/OJCOMS.2025.3529211.

[9] E. Bassetti, E. Di Paolo, F. Drago, M. Conti, and A. Spognardi, "Opening Pandora's Packet: Expose IPv6 Implementations Vulnerabilities Using Differential Fuzzing," in Applied Cryptography and Network Security, M. Fischlin and V. Moonsamy, Ed., Cham: Springer Nature Switzerland, 2025, pp. 401–423. doi: 10.1007/978-3-031-95761-1_14.

[10] C. Laskos, A. Zubow, and F. Dressler, "Latency Analysis of SDR-based

Experimental C-RAN / O-RAN Systems," in 2025 IEEE International Conference on Communications Workshops (ICC Workshops), Jun. 2025, pp. 893–898. doi: 10.1109/ICCWorkshops67674.2025.11162221.

[11] W. Feng, X. Dou, A. Taherkordi, B. Cheng, and W. Zhang, "Compresso: Latency-Aware Transmission of Compressed IoT Measurement Data Over SDN," IEEE Internet of Things Journal, vol. 12, no. 12, pp. 20462–20472, Jun. 2025, doi: 10.1109/JIOT.2025.3543479.

[12] X. Qin, R. Doss, F. Jiang, X. Qin, and B. Long, "Securing ICS networks: SDN-based Automated Traffic Control and MTD Defensive Framework against DDoS attacks," Computer Communications, vol. 241, p. 108252, Sep. 2025, doi: 10.1016/j.comcom.2025.108252.

[13] Y. Kim, T.-K. Kim, and Y. Kyung, "Efficient Resource-Aware Proactive Flow Rule Caching in Software-Defined Access Networks," in 2025 International Conference on Information Networking (ICOIN), Jan. 2025, pp. 360–362. doi: 10.1109/ICOIN63865.2025.10993105.

[14] G. U. Kumar, Prabu. U, Y. Leelakrishna, P. Siddu, and Geetha. V, "A Comparative Analysis of Traffic Analysis Frameworks in Software-Defined Networking," in 2025 International Conference on Visual Analytics and Data Visualization (ICVADV), Mar. 2025, pp. 58–62. doi: 10.1109/ICVADV63329.2025.10961511.

[15] R. Kumar et al., "End-to-End Network Delay Guarantees for Real-Time Systems Using SDN," in 2017 IEEE Real-Time Systems Symposium (RTSS), Dec. 2017, pp. 231–242. doi: 10.1109/RTSS.2017.00029.

[16] R. Amin, E. Rojas, A. Aqdus, S. Ramzan, D. Casillas-Perez, and J. M. Arco, "A Survey on Machine Learning Techniques for Routing Optimization in SDN," IEEE Access, vol. 9, pp. 104582–104611, 2021, doi: 10.1109/ACCESS.2021.3099092.

[17] P. Megyesi, A. Botta, G. Aceto, A. Pescapé, and S. Molnár, "Challenges and solution for measuring available bandwidth in software defined networks," Computer Communications, vol. 99, pp. 48–61, Feb. 2017, doi: 10.1016/j.comcom.2016.12.004.

[18] Q. Li, X. Zou, Q. Huang, J. Zheng, and P. P. C. Lee, "Dynamic Packet Forwarding Verification in SDN," IEEE Transactions on Dependable and Secure Computing, vol. 16, no. 6, pp. 915–929, Nov. 2019, doi: 10.1109/TDSC.2018.2810880.

[19] H. Farhad, H. Lee, and A. Nakao, "Data Plane Programmability in SDN," in 2014 IEEE 22nd International Conference on Network Protocols, Oct. 2014, pp. 583–588. doi: 10.1109/ICNP.2014.93.

[20] A. Bianco, R. Birke, L. Giraudo, and M. Palacin, "OpenFlow Switching: Data Plane Performance," in 2010 IEEE International Conference on Communications, May 2010, pp. 1–5. doi: 10.1109/ICC.2010.5502016.

[21] M. Elmadani and S. O. Sati, "MTU Analyzing for Data Centers Interconnected Using VxLAN," in 2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS), Jan. 2024, pp. 1825–1829. doi: 10.1109/ICETSIS61505.2024.10459403.

[22] S. Narayan and P. R. Lutui, "Network Performance Evaluation of Jumbo Frames on a Network," in 2013 6th International Conference on Emerging Trends in Engineering and Technology, Dec. 2013, pp. 69–72. doi: 10.1109/ICETET.2013.16.

[23] Q. Yuan, F. Li, T. Pan, Y. Lai, Y. Gu, and X. Wang, "INT-Segment: MTU-Adaptive Single-Path In-Band Network-Wide Telemetry," in 2022 IEEE 30th International Conference on Network Protocols (ICNP), Oct. 2022, pp. 1–11. doi: 10.1109/ICNP55882.2022.9940397.

[24] J. Zurawski, E. Kissel, G. Robb, C. Eichelberger, and N. Mendoza, "Improving Data Mobility Through Modern Security Infrastructure," in 2025 International Conference on Computing, Networking and Communications (ICNC), Feb. 2025, pp. 57–61. doi: 10.1109/ICNC64010.2025.10993877.

[25] S.-Y. Wang, "Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet," in 2014 IEEE Symposium on Computers and Communications (ISCC), Jun. 2014, pp. 1–6. doi: 10.1109/ISCC.2014.6912609.

[26] T. Jafarian, A. Ghaffari, A. Seyfollahi, and B. Arasteh, "Detecting and mitigating security anomalies in Software-Defined Networking (SDN) using Gradient-Boosted Trees and Floodlight Controller characteristics," Computer Standards & Interfaces, vol. 91, p. 103871, Jan. 2025, doi: 10.1016/j.csi.2024.103871.

[27] T. Khudhair and O. Athab, "Recent Tools of Software-Defined Networking Traffic Generation and Data Collection," Al-Khwarizmi Engineering Journal, vol. 21, no. 2, pp. 93–105, Jun. 2025, doi: 10.22153/kej.2025.06.002.

[28] R. H. Serag, M. S. Abdalzaher, H. A. E. A. Elsayed, and M. Sobh, "Software Defined Network Traffic Classification for QoS Optimization Using Machine Learning," J Netw Syst Manage, vol. 33, no. 2, p. 41, Feb. 2025, doi: 10.1007/s10922-025-09911-6.

[29] D. Bishla and B. Kumar, "Optimizing the Performance of Hybrid Software Defined Network (SDN) Through Metaheuristic Algorithms," in 2025 10th International Conference on Signal Processing and Communication (ICSC), Feb. 2025, pp. 700–709. doi: 10.1109/ICSC64553.2025.10968197.

[30] F. Syanofri and E. R. Dalimunthe, "Comparative Analysis of Quality of Service (QoS) in Wired and Wireless Networks Using Wireshark," no. 03.