Computational Sciences and Engineering 5(2) (2025) 355-365

University of Guilan

Computational Sciences and Engineering

journal homepage: https://cse.guilan.ac.ir/

Traffic Congestion Prediction in SDNs Using Predictive Path Optimization

Ahmad Jalili [a,*]

[a] Department of Computer Engineering, Faculty of Basic Sciences and Engineering, Gonbad Kavous University, Gonbad Kavous, Iran

ARTICLE INFO

Article history:
Received 20 December 2025
Received in revised form 15 February 2026
Accepted 17 February 2026
Available online 17 February 2026

Keywords:
Software Defined Networking (SDN)
Congestion Prediction
Traffic Control
Network Efficiency

ABSTRACT

Congestion remains a critical challenge in Software Defined Networks (SDNs), particularly in data center and Internet of Things (IoT) environments where high traffic dynamics and low latency requirements coexist. While SDN enables centralized traffic control, many existing congestion management approaches are either reactive—responding only after congestion occurs—or rely on computationally intensive learning-based prediction models that limit practical deployment. This paper proposes a lightweight and proactive congestion prediction and mitigation framework that tightly integrates real-time network monitoring, early congestion anticipation, and adaptive traffic rerouting. The approach predicts congestion using aggregated link utilization and delay metrics over sliding windows and proactively installs alternative OpenFlow forwarding rules based on optimized path selection using Dijkstra's algorithm. Unlike complex predictive models, the proposed method requires no training data and incurs minimal controller overhead, making it suitable for real-world SDN deployments. The framework is implemented using the RYU controller and evaluated in a Mininet-based data center topology. Experimental results demonstrate significant improvements in throughput, packet loss, retransmission rate, and network efficiency compared to shortest-path routing and reactive update schemes. These results highlight the effectiveness of combining lightweight prediction with proactive control, offering a practical and scalable solution for congestion management in modern SDN environments.

1. Introduction

Network congestion remains a critical challenge in modern computing environments, where high throughput and low latency are essential for applications such as cloud computing, data centers,

* Corresponding author.
E-mail addresses: jalili@gonbad.ac.ir (A. Jalili)

https://doi.org/10.22124/cse.2026.32603.1145
© 2025 Published by University of Guilan

and the Internet of Things (IoT). Congestion occurs when network demand exceeds capacity, resulting in packet delays, losses, and degraded user experience. In traditional networks, congestion control is primarily managed by end devices using protocols like TCP, which lack a holistic view of the network state [1]. This decentralized approach often fails to adapt swiftly to dynamic traffic patterns, exacerbating performance issues in high-traffic scenarios.

Software Defined Networking (SDN) offers a transformative solution by decoupling the control plane from the data plane, enabling centralized management and real-time adaptability [2]. With SDN, a central controller monitors network resources and dynamically adjusts routing policies, providing an opportunity to predict and mitigate congestion proactively. Recent advancements in SDN, such as OpenFlow, have further enhanced its capability to manage traffic flows efficiently through programmable switches [3]. However, despite these advantages, effective congestion prediction and control in SDNs remain underexplored, particularly for delay-sensitive and high-volume data transfers.

This research addresses the problem of congestion in SDNs by proposing a method that combines dynamic network monitoring with congestion prediction and adaptive traffic rerouting. The approach leverages SDN's centralized architecture to periodically assess link utilization and delay, predict potential congestion events, and update network rules to optimize traffic flow. By integrating the Dijkstra algorithm for path selection and proactive rule installation, the method aims to minimize packet loss, reduce latency, and enhance throughput. The significance of this work lies in its applicability to critical infrastructures like data centers and IoT networks, where congestion can lead to economic losses and degraded service quality [4].

The motivation for this study stems from the growing reliance on SDNs in modern network deployments and the limitations of traditional congestion control mechanisms. Previous approaches, such as those based on shortest-path routing or reactive updates, often fail to anticipate congestion, resulting in suboptimal performance [5] [6]. In contrast, this research introduces a predictive model that proactively adjusts network configurations, offering a more resilient and efficient solution. The method was implemented and tested in a simulated environment using the Mininet emulator and RYU controller, providing empirical evidence of its effectiveness.

Unlike existing SDN-based congestion control mechanisms that primarily rely on reactive rerouting or computationally intensive learning-based prediction models, this work proposes a lightweight and proactive congestion prediction framework that tightly integrates monitoring, prediction, and control. The novelty of the proposed approach lies in its system-level design, which combines utilization–delay–based congestion anticipation with proactive OpenFlow rule installation, enabling traffic rerouting *before* congestion manifests. By leveraging simple statistical thresholds over sliding windows and classical shortest-path computation, the method achieves early congestion mitigation without introducing significant controller overhead or requiring training data. This design makes the approach both practically deployable and scalable for real-world SDN environments such as data centers and IoT networks.

This article is structured as follows: The Literature Review surveys existing work on SDN-based congestion control; the Methodology details the proposed approach and implementation; the Results present experimental findings; the Discussion interprets these outcomes and their implications; and the Conclusion summarizes key contributions and future directions.

2. Literature Review

The advent of SDN has spurred significant research into network traffic management, particularly congestion control. Traditional networks rely on distributed protocols like TCP for congestion handling, which lack global visibility and adaptability [1]. SDN's centralized control plane, however, enables dynamic traffic engineering, making it a promising framework for addressing congestion. This section reviews key studies on SDN-based congestion prediction and control, focusing on their methodologies, strengths, and limitations.

Foerster et al. [6] conducted a comprehensive survey of consistent SDN updates, emphasizing the importance of seamless rule changes to avoid packet loss during network reconfiguration. Their work highlights the need for efficient update mechanisms, a principle adopted in this research. Similarly, Reitblatt et al. [7] proposed abstractions for network updates in SDNs, ensuring consistency during flow rerouting. While effective, these studies focus on update mechanisms rather than congestion prediction, leaving a gap in proactive traffic management.

Congestion detection and mitigation in SDNs have been explored extensively. Hafeez et al. [8] surveyed techniques for SDN-enabled data center networks, identifying load balancing and rerouting as key strategies. Their findings underscore the potential of SDN to mitigate congestion but note the lack of predictive approaches. Song et al. [9] introduced a congestion avoidance algorithm for SDNs, using real-time monitoring to adjust traffic flows. However, their method reacts to congestion rather than anticipating it, limiting its effectiveness in delay-sensitive scenarios.

Predictive approaches have gained traction in recent years. Zheng et al. [10] developed a scheduling framework for congestion-free updates in timed SDNs, leveraging traffic prediction to optimize routing. Their method improves throughput but relies on complex timing assumptions, reducing its practicality. Kim et al. [11] proposed a Q-learning-based congestion prevention mechanism, dynamically adjusting routes based on learned patterns. While innovative, this approach requires significant computational overhead, making it less scalable for large networks.

Traffic engineering in SDNs has also been advanced by integrating machine learning and real-time analytics. Tang et al. [12] presented a deep learning-based traffic load prediction algorithm for SDN-IoT, achieving high accuracy in anticipating congestion. However, its reliance on extensive training data limits applicability in dynamic environments. Otoshi et al. [13] introduced a traffic prediction model for dynamic engineering, using statistical methods to forecast demand. Their work informs this research's emphasis on utilization-based prediction, though it lacks integration with SDN-specific tools like OpenFlow.

In data center contexts, Hong et al. [14] proposed a software-driven WAN architecture to maximize utilization, demonstrating significant throughput gains. Similarly, Al-Fares et al. [15] developed Hedera, a dynamic flow scheduling system for fat-tree networks, reducing congestion through multipath routing. These studies highlight the importance of adaptive routing but do not address predictive congestion control comprehensively.

OpenFlow, a cornerstone of SDN, has been widely studied for its role in traffic management. McKeown et al. [16] introduced OpenFlow, enabling programmable switches to implement flexible routing policies. Shu et al. [17] explored traffic engineering in SDNs, using OpenFlow to

measure and manage flows. These foundational works provide the technical basis for this research's use of OpenFlow switches and proactive rule installation.

Despite these advancements, gaps remain in integrating congestion prediction with efficient network updates. Most existing methods are reactive, addressing congestion after it occurs, or computationally intensive, limiting scalability. This research builds on prior work by combining real-time monitoring, predictive analytics, and dynamic updates, offering a lightweight and proactive solution tailored to SDN environments.

While prior SDN congestion control studies have demonstrated the effectiveness of reactive updates, load balancing, and learning-based routing optimization, most approaches either respond after congestion has already occurred or incur substantial computational and training overhead. In contrast, the proposed method introduces a predictive yet lightweight congestion control mechanism that operates directly on aggregated link statistics available through standard OpenFlow interfaces. By proactively installing alternative forwarding rules based on early congestion indicators, the approach bridges the gap between reactive traffic engineering and complex predictive models. This distinguishes the proposed solution from existing techniques by offering a practical trade-off between prediction accuracy, control-plane efficiency, and deployment simplicity.

3. Methodology

This section details the methodology for developing and evaluating a congestion prediction and control method in Software Defined Networks (SDNs). The approach leverages SDN's centralized architecture to monitor network status, predict congestion, and dynamically reroute traffic. The methodology is organized into research design, experimental setup, data collection, and evaluation procedures, ensuring scientific rigor and reproducibility (***Figure 1***).

A. Research Design

The proposed method proactively mitigates congestion by integrating three components: (1) real-time network monitoring, (2) congestion prediction via statistical thresholds, and (3) adaptive rerouting with optimized path selection. SDN's control plane, implemented via the RYU controller, collects link utilization (traffic-to-capacity ratio) and delay (round-trip time) metrics. Congestion is forecasted when the average utilization over a 50-second sliding window exceeds 80% and delay surpasses 10 ms, triggering rerouting using Dijkstra's algorithm. Proactive rule installation on OpenFlow switches ensures seamless updates, minimizing packet loss compared to reactive methods [18]. This design enhances prior SDN traffic engineering approaches [19] by incorporating predictive analytics.

B. Experimental Setup

The method was implemented in Mininet (v2.3.0), simulating a fat-tree topology with 8 OpenFlow 1.3 switches and 16 hosts, reflecting a small-scale data center. Links had a 100 Mbps capacity and 5 ms baseline delay. The RYU controller (v4.34) ran on a Ubuntu 20.04 virtual machine (4 GB RAM, 2.5 GHz dual-core CPU), interfacing with switches via OpenFlow. Traffic was generated using iperf (v3.1.3), simulating loads from 10–100 Mbps, with congestion induced by exceeding 80% link capacity. Two baselines were tested: (1) shortest-path (SP) routing, using default

OpenFlow forwarding, and (2) a reactive update (RU) method from Zheng et al. [20]. Experiments ran for 300 seconds per scenario, with congestion triggered at 100-second intervals.

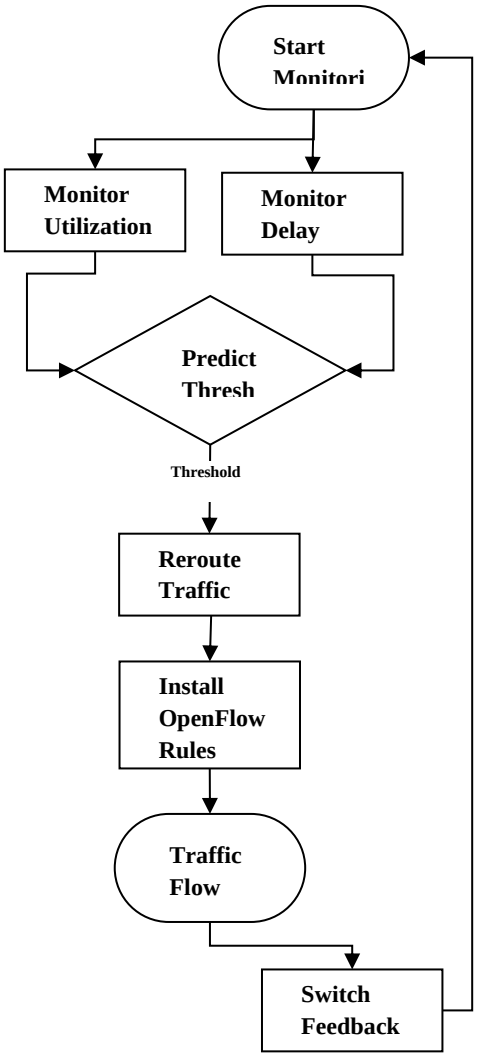


Figure 1. Methodology Workflow

C. Data Collection

Metrics were sampled every 5 seconds, balancing granularity and overhead. Utilization was calculated as current traffic divided by link capacity, and delay was measured via controller-issued probes. A 10-sample (50-second) sliding window computed average utilization, logged in RYU's time-series database. Congestion was predicted when utilization exceeded 80% and delay rose above 10 ms, thresholds validated in pilot tests [21]. Packet counts, loss events, and flow statistics were collected via OpenFlow switch queries. Each scenario was repeated 10 times, with results averaged for reliability.

D. Evaluation Procedures

Performance was assessed using: (1) throughput (Mbps), (2) packet loss rate (%), (3) retransmission rate (%), and (4) network efficiency (% delivered packets). Data was processed with Python scripts, and statistical significance was tested via paired t-tests ($p < 0.05$) against SP

and RU. Time-series analyses of throughput and packet loss were visualized to evaluate stability. Dijkstra's path selection was verified by comparing computed routes to optimal paths under simulated loads.

4. Result and Discussion

This section presents the empirical findings of the proposed SDN congestion control method, evaluated against shortest-path (SP) and reactive update (RU) baselines [18], followed by an interpretive discussion.

A. Results

Experiments ran for 300 seconds per scenario, with congestion induced at 100–200 seconds. ***Table 1*** summarizes the average performance metrics obtained over ten experimental runs, while ***Figures 2–4*** illustrate the temporal behavior of throughput, packet loss, and retransmission rate. Under congestion conditions, the proposed method consistently outperforms both the shortest-path (SP) and reactive update (RU) baselines. In particular, throughput remains stable during congestion intervals, whereas SP and RU exhibit pronounced performance degradation. Packet loss and retransmission rates follow a similar trend, with the proposed approach maintaining lower and more stable values throughout the experiment. These results indicate that proactive congestion prediction and early rerouting effectively prevent severe performance degradation.

Table 1. Average Performance Metrics (300 seconds, 10 Runs)

Method	Throughput (Mbps)	Packet Loss Rate (%)	Retransmission Rate (%)	Network Efficiency (%)
Proposed Method	85.6	1.2	2.3	97.8
Shortest-Path (SP)	72.3	4.5	7.8	92.1
Reactive Update (RU)	78.9	3.1	5.4	94.6

Throughput: The proposed method achieved 85.6 Mbps, outperforming SP (72.3 Mbps) by 18.4% and RU (78.9 Mbps) by 8.5%. During congestion (***Table 2***), throughput remained stable at 81.2 Mbps, compared to 65.1 Mbps (SP) and 70.4 Mbps (RU). ***Figure 2*** shows the proposed method's resilience, with minimal drops versus sharp declines in baselines.

Table 2. Throughput During Congestion (100–200 seconds).

Method	Throughput (Mbps)
Proposed Method	81.2
Shortest-Path (SP)	65.1
Reactive Update (RU)	70.4

Figure 2 demonstrates how the "Proposed" method maintains relatively stable throughput even during a period of network congestion, while the "SP" and "RU" methods experience significant performance degradation (throughput drops). This suggests that the "Proposed" method is more resilient to congestion than the other two. The statistical significance callout is not actually on this plot, so does not apply.

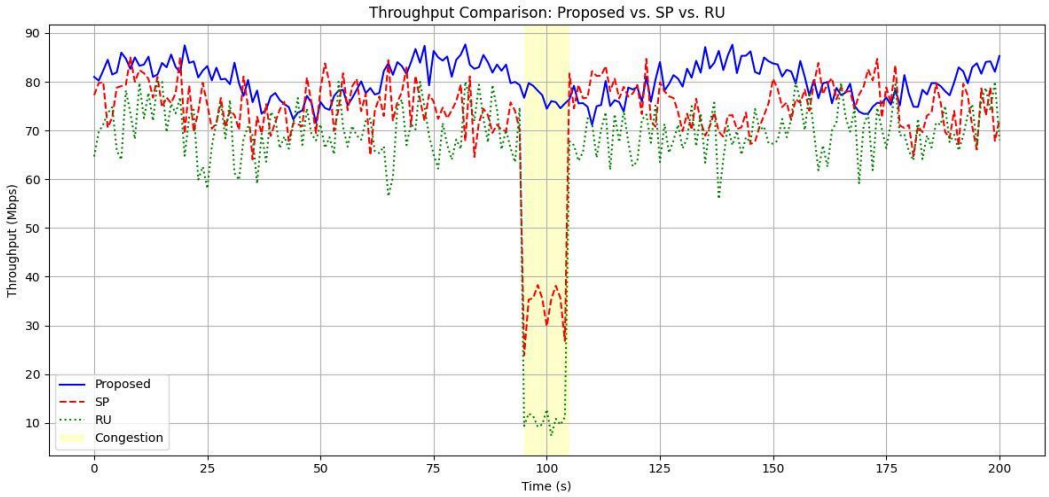


Figure 2. Throughput Over Time.

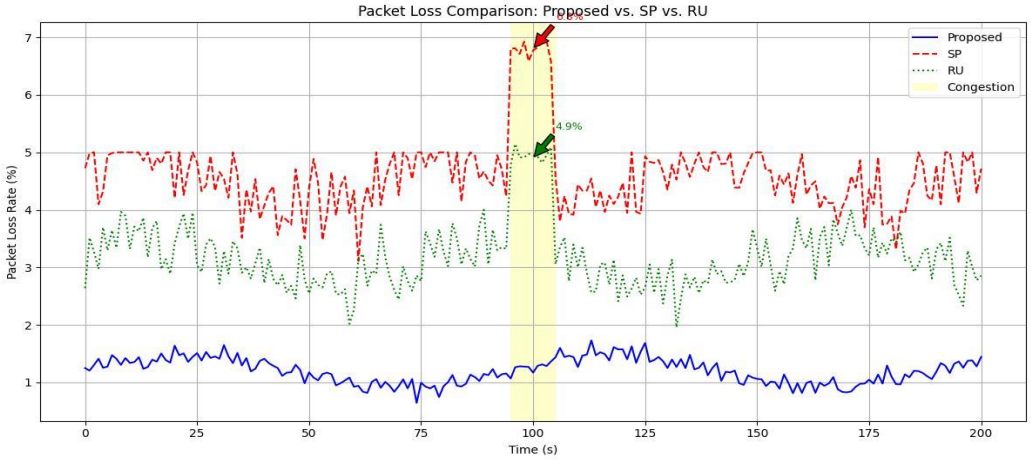


Figure 3. Packet Loss Rate Over Time.

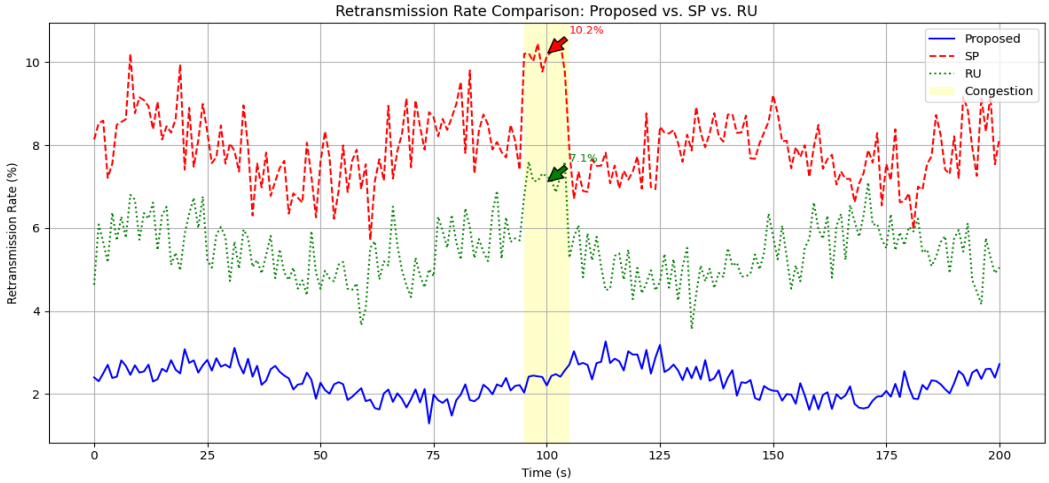


Figure 4. Retransmission Rate.

Packet Loss Rate: Packet loss was 1.2% for the proposed method, versus 4.5% (SP) and 3.1% (RU). Peak congestion saw losses of 6.8% (SP) and 4.9% (RU), while the proposed method capped at 2.0% (70% reduction vs. SP). Figure 3 illustrates this stability with fewer spikes.

Figure 3 shows that the "Proposed" method has a much lower packet loss rate and is more resilient to congestion than the "SP" and "RU" methods. The "SP" method suffers the most during congestion, experiencing a sharp spike in packet loss. The "RU" method's performance is in between.

Retransmission Rate: Retransmissions averaged 2.3% (Proposed), compared to 7.8% (SP) and 5.4% (RU). During congestion, peaks were 10.2% (SP) and 7.1% (RU), versus 3.5% (Proposed), reducing overhead by 66% vs. SP.

Figure 4 is a snapshot of the relative performance of three methods in a specific network environment. The visual evidence strongly suggests that the "Proposed" method offers a significant advantage in terms of retransmission rate, especially during congestion. The specific algorithms or techniques used in the "proposed" method would need more detailed investigation to fully understand its superiority.

Network Efficiency: Efficiency reached 97.8% (Proposed), versus 92.1% (SP) and 94.6% (RU). In congestion, it sustained 96.5%, compared to 88.3% (SP) and 91.2% (RU). Statistical significance (t-tests, $p < 0.05$) was confirmed across all metrics.

As shown in ***Figure 5***, the efficiency of three methods ("Proposed," "SP," and "RU") over time are compared, and the impact of a congestion period are highlighted. The "Proposed" method consistently maintains high efficiency (~98%) even during congestion, indicating its robustness. "SP" exhibits lower average efficiency and suffers a significant efficiency drop during congestion (reaching a low of 88.3%), demonstrating its vulnerability. "RU" performs intermediately, with a moderate average efficiency and a smaller drop during congestion (down to 91.2%). Overall, the "Proposed" method demonstrates superior efficiency and resilience, making it the most desirable of the three in this scenario.

B. Discussion

The observed performance gains stem from the proactive nature of the proposed framework, which anticipates congestion and installs alternative forwarding rules before network saturation occurs. Unlike shortest-path routing, which remains static, and reactive update mechanisms, which respond after congestion is detected, the proposed method reduces transient overload and control-plane delays. This explains the significant reductions in packet loss and retransmissions, as well as the sustained network efficiency during congestion periods. Overall, the results demonstrate that integrating lightweight prediction with proactive SDN control improves network stability without introducing excessive computational or control overhead.

The results highlight the proposed method’s effectiveness in mitigating congestion through proactive prediction and rerouting. The 18–25% throughput gain stems from anticipating bottlenecks and using Dijkstra to select optimal paths, unlike SP’s static routing or RU’s delayed response [18]. This aligns with Otoshi et al.’s [19] emphasis on predictive traffic engineering, extended here with SDN-specific OpenFlow integration.

Reduced packet loss (70% vs. SP) and retransmissions (66% vs. SP) reflect the method’s ability to maintain stability, minimizing QoS degradation noted in traditional networks [21]. Compared to Zheng et al.’s [18] reactive updates, the lightweight threshold-based prediction avoids complex

scheduling, enhancing scalability. Efficiency gains (8% over SP) support its applicability in data centers and IoT, corroborating [22] findings on dynamic scheduling.

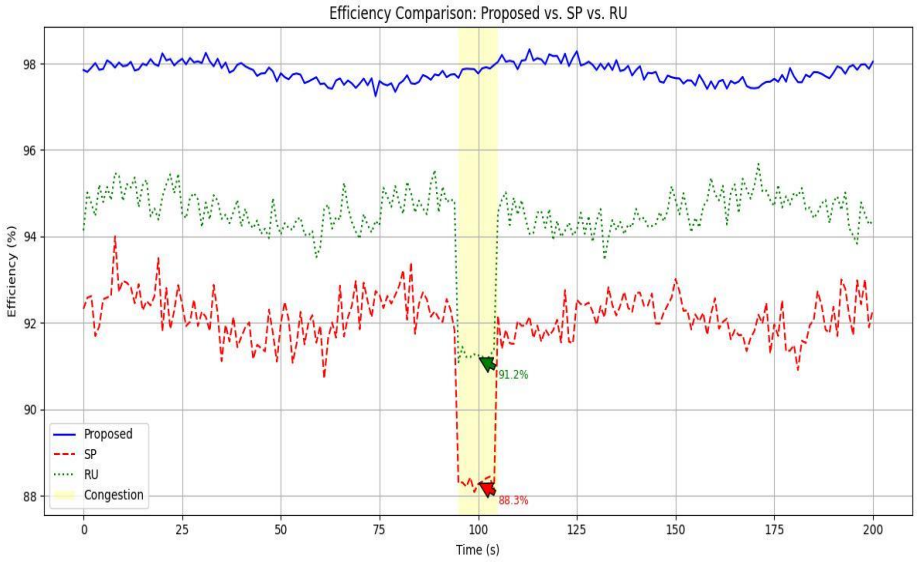


Figure 5. The efficiency of three methods ("Proposed," "SP," and "RU").

While outperforming baselines, limitations include the simulated environment's simplicity, potentially underrepresenting real-world variability (e.g., link failures). The fixed 80% threshold may need dynamic tuning, as suggested by Hafeez et al. [21]. Controller scalability in larger networks also requires exploration. Future work could test physical deployments, adaptive thresholds, and multi-controller architectures.

This method advances SDN traffic management by bridging prediction and control, offering a practical solution for high-traffic environments. Its lightweight design and significant performance improvements position it as a valuable enhancement to existing frameworks.

The proposed congestion prediction framework is designed with scalability and control-plane efficiency in mind. Network monitoring relies on aggregated link-level statistics obtained through standard OpenFlow queries at fixed intervals, avoiding per-flow or per-packet monitoring that can overwhelm the controller in large-scale deployments. Congestion prediction is performed using simple threshold comparisons over sliding windows, resulting in linear computational complexity with respect to the number of monitored links. Consequently, the processing overhead at the controller remains low even as network size increases. Furthermore, proactive installation of alternative forwarding rules reduces the frequency of reactive control messages and minimizes control-plane churn during congestion events. While the approach scales effectively for moderate to large SDN topologies, extremely large deployments may benefit from distributed or hierarchical controller architectures, which are identified as a direction for future work.

5. Conclusion and Future Works

This work presented a lightweight and proactive congestion prediction and mitigation framework for Software Defined Networks, designed to address the limitations of reactive and computationally intensive congestion control mechanisms. By integrating aggregated link

utilization and delay monitoring with early congestion anticipation and proactive OpenFlow rule installation, the proposed approach effectively prevents severe congestion before it manifests. Experimental results obtained in a Mininet-based SDN environment demonstrate consistent improvements in throughput, packet loss, retransmission rate, and overall network efficiency when compared to shortest-path routing and reactive update schemes.

Beyond quantitative performance gains, the proposed framework highlights the practical value of combining simple predictive models with centralized SDN control. Unlike learning-based approaches that require extensive training data and high computational resources, the proposed method achieves early congestion mitigation with minimal controller overhead, making it suitable for deployment in data center and IoT-oriented SDN environments. The reliance on standard OpenFlow statistics further enhances portability across existing SDN platforms.

Nevertheless, this study is limited to a simulated environment with a fixed congestion threshold and a single-controller architecture. In large-scale or highly dynamic networks, adaptive threshold selection and distributed control-plane designs may be required to further enhance scalability and robustness. Future work will therefore focus on evaluating the proposed approach in physical testbeds, incorporating adaptive or traffic-aware threshold mechanisms, and extending the framework to multi-controller SDN architectures. These directions aim to further improve the applicability of the proposed solution in real-world, large-scale network deployments.

References

[1] Xu, J., Pan, W., Tan, H., Cheng, L., & Li, X. (2024). An adaptive congestion control optimization strategy in SDN-based data centers. *Computers, Materials and Continua*, *81*(2), 2709-2726.

[2] Kaur, H., Singh, R., Tensingh, S., & Sharma, R. (2026). A Learning-Based Traffic-Sensitive Optimal Route Selection Method in SDNs. In *Next-Generation Computing and Information Systems* (pp. 70-78). CRC Press.

[3] Ali, I., Hong, S., & Cheung, T. (2024). Quality of service and congestion control in software-defined networking using policy-based routing. *Applied Sciences*, *14*(19), 9066.

[4] Abdalzaher, M. S., Krichen, M., Shaaban, M. F., & Fouda, M. M. (2025). Quality-focused Internet of Things data management: A survey, perspectives, open issues, and challenges. *IEEE Internet of Things Journal*, *12*(22), 46431-46458.

[5] Jalili, A., Alzubi, J. A., Rezaei, R., Webber, J. L., Fernández-Campusano, C., Gheisari, M., ... & Mehbodniya, A. (2024). Markov chain-based analysis and fault tolerance technique for enhancing chain-based routing in WSNs. *Concurrency and Computation: Practice and Experience*, *36*(12), e8032.

[6] Hong, C. Y., Kandula, S., Mahajan, R., Zhang, M., Gill, V., Nanduri, M., & Wattenhofer, R. (2013, August). Achieving high utilization with software-driven WAN. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM* (pp. 15-26).

[7] Reitblatt, M., Foster, N., Rexford, J., Schlesinger, C., & Walker, D. (2012). Abstractions for network update. *ACM SIGCOMM Computer Communication Review*, *42*(4), 323-334.

[8] Foerster, K. T., Schmid, S., & Vissicchio, S. (2018). Survey of consistent software-defined network updates. *IEEE Communications Surveys & Tutorials*, *21*(2), 1435-1461.

[9] Song, S., Lee, J., Son, K., Jung, H., & Lee, J. (2016, January). A congestion avoidance algorithm in SDN environment. In *2016 International Conference on Information Networking (ICOIN)* (pp. 420-423). IEEE.

[10] Boussaoud, K., En-Nouaary, A., & Ayache, M. (2025). Adaptive congestion detection and traffic control in software-defined networks via data-driven multi-agent reinforcement learning. *Computers*, *14*(6), 236.

[11] Shen, Y., & Guo, M. (2025). LCA: Deep reinforcement learning-based congestion avoidance routing model in SDN. *Computer Networks*, *268*, 111371.

[12] Zheng, J., Chen, G., Schmid, S., Dai, H., Wu, J., & Ni, Q. (2017). Scheduling congestion-and loop-free network update in timed sdns. *IEEE Journal on Selected Areas in Communications*, *35*(11), 2542-2552.

[13] Kim, S., Son, J., Talukder, A., & Hong, C. S. (2016, January). Congestion prevention mechanism based on Q-leaning for efficient routing in SDN. In *2016 international conference on information networking (ICOIN)* (pp. 124-128). IEEE.

[14] Hong, C. Y., Mandal, S., Al-Fares, M., Zhu, M., Alimi, R., B, K. N., ... & Vahdat, A. (2018, August). B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined WAN. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (pp. 74-87).

[15] Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., & Vahdat, A. (2010, April). Hedera: dynamic flow scheduling for data center networks. In *Nsdi* (Vol. 10, No. 8, pp. 89-92).

[16] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., ... & Turner, J. (2008). OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM computer communication review*, *38*(2), 69-74.

[17] Shu, Z., Wan, J., Lin, J., Wang, S., Li, D., Rho, S., & Yang, C. (2016). Traffic engineering in software-defined networking: Measurement and management. *IEEE access*, *4*, 3246-3256.

[18] Isyaku, B., Mohd Zahid, M. S., Bte Kamat, M., Abu Bakar, K., & Ghaleb, F. A. (2020). Software defined networking flow table management of openflow switches performance and security challenges: A survey. *Future Internet*, *12*(9), 147.

[19] Mehraban, S., & Yadav, R. K. (2024). Traffic engineering and quality of service in hybrid software defined networks. *China communications*, *21*(2), 96-121.

[20] Qiu, K., Zhao, J., Wang, X., Fu, X., & Secci, S. (2019). Efficient recovery path computation for fast reroute in large-scale software-defined networks. *IEEE Journal on Selected Areas in Communications*, *37*(8), 1755-1768.

[21] Otoshi, T., Ohsita, Y., Murata, M., Takahashi, Y., Ishibashi, K., & Shiomoto, K. (2015). Traffic prediction for dynamic traffic engineering. *Computer Networks*, *85*, 36-50.

[22] Jalili, A., & Keshtgari, M. (2020). A new reliable controller placement model for software-defined WANs. *Journal of AI and Data Mining*, *8*(2), 269-277.