Jurnal Kejuruteraan 38(2) 2026: 753-765
https://doi.org/10.17576/jkukm-2026-38(2)-23

Performance Optimization in SD-WAN Using Bucket-Based Packet Aggregation Algorithm

Muhammad Haqeem Mohd Nasir[a] & Fairuz Abdullah[b*]

[a]*College of Engineering, Universiti Tenaga Nasional*

[b]*Institute of Power Engineering, Universiti Tenaga Nasional*

[*]*Corresponding author: fairuz@uniten.edu.my*

Received 17 June 2025, Received in revised form 24 November 2025
Accepted 24 December 2025, Available online 30 March 2026

ABSTRACT

Wide Area Network (WAN) play a vital role in contemporary network systems, facilitating communication among devices across broad geographic regions. Despite its importance, WAN infrastructure often comes with high complexity. To address this issue, Software-Defined Networking (SDN) has evolved as a means to decouple the control logic from its data forwarding components, simplifying the overall network architecture and facilitating more streamlined management. Combining SDN and WAN results in SD-WAN, a more efficient and cost-conscious approach that delivers enhanced Quality of Service (QoS). However, as technology continues to advance, the volume of network traffic grows significantly that leads to congestion because each packet is processed individually. Furthermore, Customer Premise Edge (CPE) in SD-WAN becomes the bottleneck of the network makes the congestion worse. Most of recent studies rely on complex control mechanisms and process packets individually. These limitations highlight the need for a lightweight, proactive strategy that reduces congestion by decreasing packet-processing frequency. Hence, this paper proposes a novel congestion control algorithm based on the bucket sort technique that aggregates packets with identical destination network IPs into a single bucket, then, sending them together as one packet to reduce processing load. We simulated the proposed algorithm using Python and compared it against a standard loss-based congestion control method in GNS3. The results demonstrate that our proposed algorithm achieves higher throughputs which are 13,882 kbps in 10 Gbps link and 27,764 kbps in 20 Gbps link. Hence, validating its effectiveness and showing its potential to enhance network performance under heavy traffic conditions.

Keywords: SDN, SD-WAN, TCP, congestion control, bucket sort

INTRODUCTION

Wide area network (WAN) is a telecommunication network interconnecting multiple Local Area Networks (LANs) that are distributed across different geographic areas. Throughout the years, many different technologies emerged with the purpose of letting enterprises to lease capacity from the operators of the public network infrastructure for building their inter-site connections. The technologies are Asynchronous Transfer Mode (ATM), Frame Relay (FR), Multiprotocol Label Switching (MPLS) and Virtual Private Networks (VPN). Currently, MPLS is the one that is widely deployed due to its promising features to provide high efficiency of Quality of Service (QoS) (Troia et al. 2020). However, the cost of MPLS for enterprise user is quite high because it requires additional layer in between of Layer 2 and Layer 3 of the OSI model (Obiodu & Sastry, 2020; Ridwan et al. 2020). On the other hand, VPN can be created on virtually any kind of underlay network and connecting enterprise sites with simple and inexpensive Internet connection. However, VPN does not provide guaranteed QoS. Hence, Software-Defined Wide Area Networking (SD-WAN) emerges to provide guaranteed high efficiency of QoS with low cost (Troia et al. 2020).

SD-WAN relies on Software-Defined Networking (SDN) that separates the physical forwarding elements and network's control logic which are called data plane and control plane respectively. This simplifies the network

control, management, and enables innovation through network programmability. The monitoring system and programmability of SD-WAN such as traffic engineering and congestion control algorithm resides in the application plane. SD-WAN reduces operational costs, streamlines configuration, and simplifies access to cloud services when compared to MPLS (Troia et al. 2020). On the other hand, compared to VPN, the control logic of SD-WAN able to provide a good degree of QoS and reliability at the same access cost of VPN. This can be achieved by having a centralized controller that resides in control plane to constantly monitors the network (Troia et al. 2020). The architecture of SD-WAN is only implemented at the edges of the network, which is called Customer Premises Edge (CPE) that reside in data plane as shown in Figure 1. The CPEs will execute the switching and routing functions based on the decision made by control plane (Troia et al. 2020).

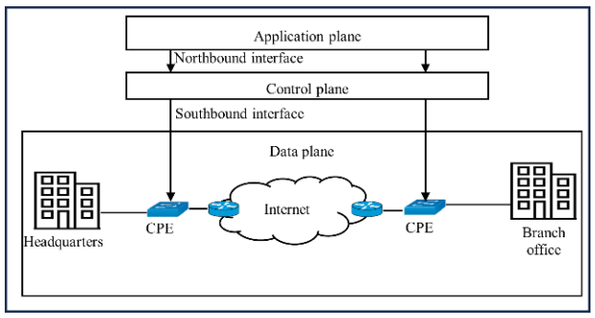

FIGURE 1. SD-WAN architecture

Additionally, SD-WAN can utilize diverse transport technologies, including 4G and 5G, to improve overall network efficiency (George et al. 2023). Nowadays, the world is relying on 5G as the main transport technology where it allows a large range of application scenarios and efficient spreading to the concept of "everything is connected". 5G delivers Ultra-Reliable Low Latency Communication (URLLC), supporting faster responsiveness than earlier technologies, with target latencies of 1 millisecond for both uplink and downlink (Lorincz et al. 2021; Zebari et al. 2021) It supports a lot of services and applications in two scenarios. The first one is wireless communication and machine-to-machine communication services. The second one is related to supporting the big data services such as streaming 4K quality videos, wireless sensor network, surveillance system, and smart grids (Zebari et al. 2021). In order to ensure the reliability of end-to-end connections in 5G network, Transmission Control Protocol (TCP) is the primary communication protocol employed in modern networks, and its congestion control mechanism plays a key role in maintaining performance (Lorincz et al. 2021).

However, modern communication network needs to accommodate data rates up to 20 Gbps and 10 Gbps for downlink and uplink respectively (Lorincz et al. 2021) making the traditional congestion control might not be enough. Furthermore, CPE that serves as the on-premises device that connects local networks to broader SD-WAN infrastructure can become overwhelmed by high data loads. Conventional Customer Premises Equipment (CPE) often possess limited processing capabilities and bandwidth, rendering them vulnerable to congestion during spikes in data traffic (Ouamri et al. 2025). When these devices are unable to efficiently manage inbound and outbound data flows, they become bottlenecks within the network, resulting in increased latency, packet loss, and overall performance degradation.

The problem worsens when the environment has multiple high-bandwidth applications are in use. For example, in businesses that rely on cloud services, video conferencing, and real-time data analytics, the demand for data throughput can easily exceed the CPE's capacity. When this happens, the CPE struggles to prioritize and route traffic effectively, causing network congestion and performance issues (Ouamri et al. 2025). This bottleneck not only affects data transmission speeds but also impacts the quality of critical services, leading to potential disruptions in business operations and reduced productivity. As the limitations of CPE in SD-WAN environments continue to contribute to network congestion, there is a pressing need for innovative solutions to address this issue.

To tackle this issue, this paper proposes a bucket sort algorithm as a congestion control that is specifically designed to reduce congestion and improve the throughput between CPEs. The bucket algorithm is implemented in the switches that reside in the CPE as they are programmable. Bucket sort is used in this proposed algorithm because it is useful when sorting elements that are uniformly distributed over an interval. Furthermore, this sorting algorithm is useful for cases where the range of input values are small (Song & Li, 2023). In this case, the elements are the individual packets from the senders. At first, the senders send 25 bytes for 30 seconds. Then, the packet size is increased by 25 bytes for each interval.

The switch at the sender side receives multiple packets simultaneously. Then, it inspects each packet's destination network Internet Protocol (IP) address. If the packets have the same destination network IP address, it puts in the same bucket. Once the bucket is full, the switch sends this bucket as one packet. When the routers in the network receives this bucket, they perceive the bucket as one packet resulting in reduced congestion because the routers have memory to process other packets. Once the switch at the receiver side receives the bucket, it sorts out all the contents and sends the individual packet to the receiver. The details of

proposed algorithm will be explained in the Methodology section.

This outline of this paper is as follows; literature review, methodology, results, discussion, and conclusion.

LITERATURE REVIEW

In this section, we will present the state-of-the-art methods presented by other researchers in reducing congestion in SD-WAN. For example, authors in (Alwasel et al. 2020), the authors present IoTSim-SDWAN to simulate the behavior and properties of SDN-WAN and SDN-enabled datacenters using a modified Dijkstra algorithm which selects links that has the minimum hop count. The proposed work provides variety of modeling approaches and functionalities to evaluate and test SD-WAN cloud-based solutions and gain additional insights on the design of future systems without requiring in-the-field experimentation that would be either prohibitively expensive or not practical in live system. In this work, the authors assigned bandwidth to each link to avoid congestion. However, the bandwidth for each link is assumed to be equal amount. Hence, this can be a bottleneck for bandwidth-hungry applications. Furthermore, they did not implement any state-of-the-art congestion control algorithm such as memory management in case of congestions.

In (Shojaee et al. 2020) present SafeGuard in SD-WAN that reduces the number of congested links by using heuristic algorithm called multi-objectives MILP optimization that improves bandwidth allocation and switch-memory usage while working on any connected network. Upon congested link, the algorithm takes the network topology as the input to set a new route for the traffic. Then, it sorts the affected traffic and available routes in descending and ascending order according to their demands and route length respectively. Then, it checks the switch's memory whether it is enough to accommodate the affected flow. One of the disadvantages of this algorithm is it computes a new route only after the congestion happens. This will lead to a lot of packet drops and reducing the throughput. Furthermore, it checks the memory availability one by one. If one of the switches unable to accommodate the flow, the controller needs to find another switch that can accommodate it. This will take a lot of time to compute especially when there is a spike of traffic in the network.

Authors in (Quang et al. 2022) propose a versatile intent-based policy optimization model which is Mixed Integer Programming (MIP) model that carefully selects the set of overlay links which are allowed for application based on their requirements and overall intents of the operator. The algorithm supports several intents such as minimization of the congestion or the maximization of the network quality. In this work, a non-linear mathematical is used to determine the best routing policy according to several global intents and Service Level Agreement (SLA) requirement applications. Furthermore, they use polynomial regression and neural network algorithm to predict the delay of the links. Hence, it adds several layers of complexity for the controller as it also needs to provide information to the prediction model while fulfilling the intents (Leivadeas & Falkner, 2022). This will overload the controller when the simulation is run for a long period of time. In the presented results, the simulation was done for only 1200 s.

Furthermore, authors in (Troia et al. 2022) present an experimental SD-WAN solution capable of running and optimizing delay-sensitive high-priority services, such as real-time video streaming, while minimizing downtime caused by network failures. They developed an ONOS application call SD-WAN Traffic Engineering (TE) to manage and improve traffic engineering at the edge of enterprise network that is supported by Transport layer Monitoring System (TPM) system. The system is in charge of measuring the number of TCP retransmission of the traffic flows and warning the network controller of possible failures. The number of TCP transmission is measured with the pre-defined threshold to check if it is exceeded or not. If it is exceeded, the controller switches the flow into a different tunnel. However, the pre-defined threshold number is an assumption where it might not be the optimal number for the traffic. Furthermore, the threshold might not be suitable all the time as the nature of traffic in WAN is dynamic, especially in 5G era where there are various types of traffics and priorities. In this generation of telecommunication, TCP retransmission is unavoidable. Hence, if the threshold is set at a low number as suggested by author, the switching and rerouting will occur too frequent and it will adversely affect the performance (Zhang et al. 2021).

Next, in (Zhang et al. 2021), the authors propose WAN-aware MPTCP that combines multiple WAN links into one "big pipe" for better WAN resilience to minimize application performance degradation under WAN link failure. They leverage the congestion control of MPTCP congestion control algorithm to balance traffic across multiple WAN links. When SDN gateway detects a path is down, it reroutes MPTCP subflows to an alternative healthy path. However, when using this method in a busy network, there is high probability of packet collision. Hence, causes even bigger congestion causing the throughput to reduced (Jowkarishasaltaneh & But, 2022). Table 1 shows the summary of the literature review.

In 2024, (Yang & Tsai 2024) propose Bottleneck Routing Feedback (BRF) algorithm which is a congestion control and bandwidth algorithm scheme for 5G network. Their method deploys an SDN controller to continuously monitor bottleneck conditions by periodically polling the switch to collect queue length, throughput, and flow statistics. The switch is modified to embed two bytes of bandwidth information into Acknowledgment (ACK) packet option fields enabling the sender to adjust its pacing rate based on real-time feedback. Simulation results in Mininet demonstrate that BRF improves bandwidth utilization, achieves faster convergence, enhances fairness, and reduces queueing delay compared to end-to-end schemes such as CUBIC. However, the reliance on frequent controller polling introduces considerable control plane overhead, increases the complexity on both the controller and switches.

In the same year, (Tong et al. 2024) propose Adaptive Bottleneck Bandwidth and Round-Trip Time (ABBR) that incorporates with Reinforcement Learning (RL). This approach formulates congestion control as a multi-armed bandit problem and applies a constraint-ε-greedy policy to balance exploration and exploitation while optimizing a reward function derived from the modified Kleinrock metric. ABBR achieves throughput comparable to BBR-Tsunami while maintaining latency close to BBR+. As with most RL-based systems, ABBR may experience unstable behavior during early exploration phases and is highly sensitive to reward design which can lead to unintended rate adjustments when network feedback fluctuates. Fairness also degrades significantly when competing with non-BBR algorithms such as CUBIC, highlighting RL's tendency to optimize its own reward at the expense of fair bandwidth sharing.

In 2025, (Watanabe & Hasegawa, 2025) propose in-network congestion control (NCC) that divides an end-to-end connection into multiple sub-paths using dedicated NCC nodes deployed within the network. Each sub-path operates its own localized congestion-control loop, allowing the use of specialized algorithms tailored to its characteristics, rather than relying solely on a single end-to-end TCP mechanism. This design improves throughput and fairness by addressing congestion closer to where it occurs. However, this approach introduces added complexity as deploying multiple NCC nodes increases coordination overhead among sub-path controllers.

In conclusion, existing SD-WAN congestion mitigation approaches span a wide range of techniques, including routing-based congestion avoidance (Alwasel et al. 2020), heuristic post-failure rerouting (Shojaee et al. 2020), policy-driven intent optimization (Quang et al. 2022), transport-layer monitoring-based rerouting (Troia et al. 2022), and multipath load balancing using MPTCP (Zhang et al. 2021). More recent advancements demonstrate a shift toward intelligent and network-aware mechanisms, such as SDN-assisted bottleneck feedback control in BRF (Yang & Tsai, 2024), RL-driven adaptive congestion control in ABBR (Tong et al. 2024), and multi-loop in-network regulation through NCC nodes (Watanabe & Hasegawa, 2025). These solutions offer notable performance improvements such as better bandwidth utilization, faster convergence, improved fairness, and adaptive rate control. However, SDN-based designs depend on frequent controller polling, RL-based methods often suffer from reward sensitivity and fairness issues, and in-network schemes require additional nodes and complex coordination across multiple control loops. Most existing approaches continue to operate at the routing, controller, and transport layers which are still processing packets individually. These approaches lead to persistent queuing, packet buildup, and high processing overhead during congestion. These limitations reveal a gap for lightweight and proactive algorithm to reduce congestion by lowering packet-processing frequency rather than reacting after congestion has formed. This gap motivates the bucket-based packet aggregation method proposed in the next section.

TABLE 1. Literature review summary

Reference	Methods	Advantages	Disadvantages
(Alwasel et al. 2020)	Uses a modified Dijkstra-based routing algorithm that selects shortest-path routes based on minimum hop count, combined with bandwidth assignment per link within their IoTSim-SDWAN simulator.	Enables evaluation of SD-WAN solutions without expensive real-world testing.	Assumes equal bandwidth for all links, unsuitable for bandwidth-hungry apps; lacks advanced congestion control.
(Shojaee et al. 2020)	Implements SafeGuard, a heuristic based on multi-objective MILP optimization that reroutes traffic after congestion by sorting flows and candidate paths according to demand and path length.	Improves bandwidth allocation and memory usage; works on any connected network.	Reacts only after congestion occurs, causing packet drops; slow route computation during traffic spikes.

continue...

...cont.

Reference	Methods	Advantages	Disadvantages
(Quang et al. 2022)	Applies an intent-based policy optimization model using MIP, with polynomial regression and neural networks for delay prediction across overlay links.	Optimizes overlay links according to multiple intents like minimizing congestion and maximizing quality.	Adds computational complexity; controller overloaded during long simulations.
(Troia et al. 2022)	Employs TPM to detect TCP retransmissions and triggers ONOS-based rerouting using predefined tunnel switching thresholds.	Minimizes downtime and improves delay-sensitive services performance.	Static retransmission threshold may cause too frequent rerouting, negatively impacting performance.
(Zhang et al. 2021)	Utilizes WAN-aware MPTCP to aggregate WAN links and reroute subflows during path failures via SDN-controlled gateways.	Enhances WAN resilience and minimizes performance degradation during link failure.	Risk of packet collision and increased congestion in busy networks, reducing throughput.
(Yang & Tsai 2024)	Propose BRF controller to continuously monitor bottleneck conditions by periodically polling the switch to collect queue length, throughput, and flow statistics. The switch is modified to embed two bytes of bandwidth information into Acknowledgment (ACK) packet option fields enabling the sender to adjust its pacing rate based on real-time feedback	BRF improves bandwidth utilization, achieves faster convergence, enhances fairness, and reduces queueing delay compared to end-to-end schemes such as CUBIC	The reliance on frequent controller polling introduces considerable control plane overhead, increases the complexity on both the controller and switches
(Tong et al. 2024)	Propose ABBR that formulates congestion control as a multi-armed bandit problem and applies a constraint-ε-greedy policy to balance exploration and exploitation while optimizing a reward function derived from the modified Kleinrock metric	ABBR achieves throughput comparable to BBR-Tsunami while maintaining latency close to BBR+.	ABBR may experience unstable behavior during early exploration phases and it is highly sensitive to RL's reward design which can lead to unintended rate adjustments when network feedback fluctuates

METHODOLOGY

To achieve the second objective of this paper, a bucket sort algorithm is used. It is a good choice for sorting elements with values uniformly distributed over an interval (Mahafzah, 2024).

In this paper, a point-to-point topology is employed as shown in Figure 2. The Integrated Development Environment (IDE) used to simulate the topology is PyCharm. It is lightweight yet powerful IDE as multiple instances can be run simultaneously. As a proof of concept, the network topology includes two CPEs, each equipped with an OpenFlow switch and three routers, labelled R1, R2, and R3 (Abane et al. 2025; Alotaibi et al. 2022; Nguyen & Debroy, 2022), and a centralized SDN controller. Three senders, which are Sender 1, Sender 2, and Sender 3, are connected to CPE 1 and a receiver is connected to CPE 2. Unlike prior work in (Troia et al. 2022), which utilizes multiple channels, this paper proposes the use of a single-channel setup to simulate a bottleneck environment where congestion is more likely to occur. The main contribution of this paper lies in the programming of switch in CPE 1 and CPE 2. A centralized controller is used in this topology to assign the IP address for each CPE.

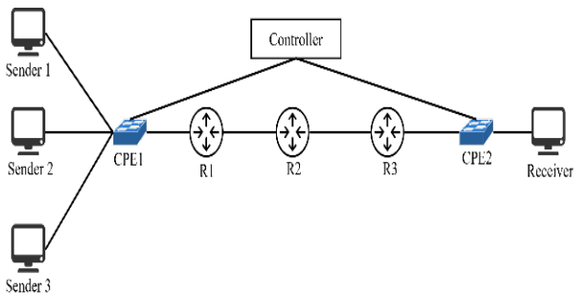

FIGURE 2. Simulation topology

To simulate the algorithm in this topology, there are four assumptions as the default system parameters. Firstly, the bucket size is configured to approximate the Maximum Transmission Unit (MTU) as closely as possible, to maximize the number of packets it can hold. The standard MTU size is typically 1500 bytes (B). (Hussain & Bashir, 2022). However, due to hardware limitations in the simulation environment, specifically, default configurations

that cannot be altered, the bucket size is constrained to 1448 B. Secondly, the packet size generated by each client is set to 25 B, with each client transmitting 30 times to obtain a reliable average for performance analysis. (Andújar et al. 2023; Sahu et al. 2023). This configuration allows a greater number of packets to be inserted into the bucket. Each packet consists of a 20 B transport header, a 20 B network header, and a 20 B data link header. (Alwasel et al. 2020). The third assumption is the bandwidth is configured at 1 Gigabit per second (Gbps), and both transmission and processing latencies are considered negligible, particularly in WAN environments, as the rate exceeds 10 Megabits per second (Mbps) (Chefrour, 2021; Tan et al. 2022). Table 2 shows the summary of the assumptions.

TABLE 2. Simulation assumptions

Items	Specifications
Bucket size	1448 B
Packet payload	25 B
Transport header	20 B
Network header	20 B
Datalink header	20 B
Bandwidth	1 Gbps

Traditionally, a packet consist of the sum of packet headers and its payload as shown in Equation (1) (Alwasel et al. 2020), where V_p is the packet size, ph is the size of packet headers and pl is the size of the payload. Hence, the size of every packet is the addition between the size of the packet header with the payload. However, by using this traditional method, the routers will check the forwarding table every time they received an individual packet. This makes other packets to build up the queue and leads to congestion.

$$V_p = ph + pl \tag{1}$$

To solve the build-up queue, Equation 1 is further derived to get the bucket size as shown in Equation (2). In Equation (2), the total size of the bucket, V_b, is the addition between the size of the bucket header, bh, with the bucket payload, bl. The bucket payload is the summation of all individual packets, V_p, from the first packet ($\alpha = 1$) until the n^{th} packet that are inserted into the bucket. The limit of the bucket payload must not exceed 1448 B. Once it reaches its limit, bucket header is added and the bucket is sent as an individual packet containing all the packets. By doing so, the routers only need to check the forwarding table once. Hence, reducing the numbers of queueing packets.

$$\begin{aligned} V_b &= bh + bl \\ &= bh + \left(\lim_{V_p \to 1448} \sum_{\alpha=1}^{n} (V_p)_\alpha \right) \\ &= bh + \left(\lim_{V_p \to 1448} \sum_{\alpha=1}^{n} (ph + pl)_\alpha \right) \\ &where\ \alpha = 1,2,3, \dots, n^{th} \end{aligned} \tag{2}$$

Once the parameters are identified, the flow of the proposed algorithm can be derived. First, the connection between nodes must be established first. The sequence of the connection is illustrated in Figure 3. The connection starts when all three senders; Sender 1, Sender 2, and Sender 3, are connected to the switch in CPE 1. In this simulation, the IP address of each sender is set to 192.168.1.101, 192.168.1.102, and 192.168.1.103 for Sender 1, Sender 2, and Sender 3 respectively. Then, the switch in CPE 1 connects with the first router which is R1. Then, R1 connects with the second router, R2. Next, R2 connects with R3. R3 connects with switch in CPE 2. The IP address of R1, R2, and R3 are initially set to 1.1.1.1, 2.2.2.2, and 3.3.3.3 respectively. The switch in CPE 2 is then connected to the receiver. The IP address of the receiver is set to 192.168.2.100. Then, the senders and receiver provide their IP addresses to their respective CPE. Finally, both of the switches connect with the controller. The CPEs are then sent the senders' and receiver's IP addresses to the controller for it to register in the forwarding tables. Then, the controller sends back the forwarding tables to the switches. The controller also assigns the switches with their own IP address. In this case, the IP addresses are 192.168.1.1 and 192.168.2.1 for CPE 1 and CPE 2 respectively. Upon receiving the forwarding tables, the senders are now ready to send the packets to the receiver. In Figure 3, the arrows with the dotted lines indicate that the switches are still waiting for the connection with the controller. Figure 4 shows the assigned IP address to each node in the network.

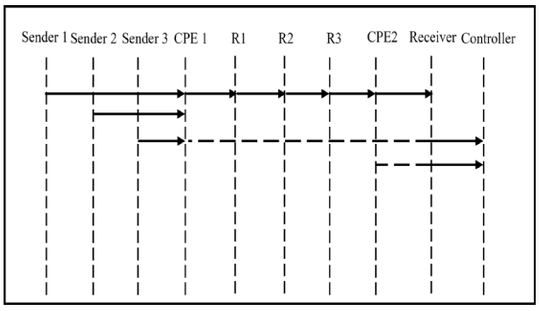

FIGURE 3. Connection sequence for all network nodes

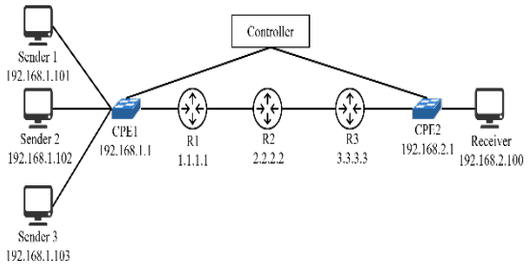

FIGURE 4. Network nodes assigned IP addresses

The senders send data simultaneously to the receiver through the CPE 1, R1, R2, R3, and CPE 2. This algorithm is not well-suited for scenarios involving a single client, as filling the bucket would require more time. Therefore, the topology incorporates three senders transmitting data concurrently as a proof of concept, enabling the bucket to fill more rapidly.

Figure 5 shows the proposed algorithm implemented in CPE 1. Once the switch in CPE 1 receives packets, it checks if they have the same destination network IP address or not by checking the first octet of IP address of each sender. In this simulation topology, the source and destination network IP addresses are 192.168.1.1 and 192.168.2.1 respectively. The algorithm uses this information as the source and destination IP addresses of the bucket. Then, the packets that have the are added into the same bucket and encapsulates it using the headers mentioned in the Table 1. If the packet does not belong to the same destination network IP address, it is sent individually. The packets inside the bucket have become the payload of the bucket as shown in Figure 6. For simplicity, we just show the header that contains the source and destination IP addresses of the bucket. Then, the algorithm calculates the size of each packet and continue appending them into the bucket until the incoming individual packet unable to fit into the bucket. Then, the switch at CPE 1 sends this bucket as one packet. The individual packet that is unable to fit into the bucket will be sent individually afterwards.

By doing this, the routers perceive this bucket as one packet. Hence, they only need to look up the forwarding table once. Compared to traditional method, the routers have to look up the forwarding table every time they receive individual packet. Thus, increasing delay and reducing overall network performance due to congestion caused by queuing packets. Figure 7 shows the sequence of forwarding the buckets and the individual packets from the senders to the receiver.

Figure 8 shows the algorithm implemented in CPE 2. Upon receiving the bucket, the switch at CPE 2 sorts out all the packets into an array by distinguishing them based on their source IP addresses. Each packet is then arranged accordingly, as shown in Figure 9. The sequence of the packets in the array depends on which packet inserted first into the bucket. By doing so, the algorithm able to know the index of each packet in the array. After all the packets are in the array, the switch sends the packets one by one to the receiver.

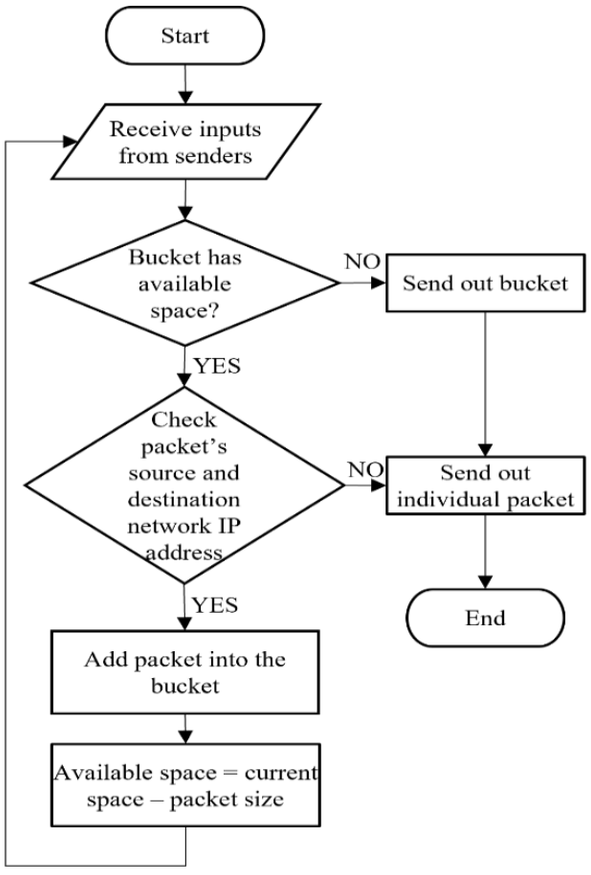

FIGURE 5. Algorithm in CPE 1

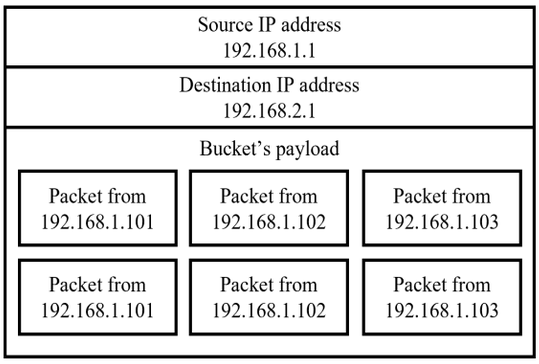

FIGURE 6. Bucket's payload

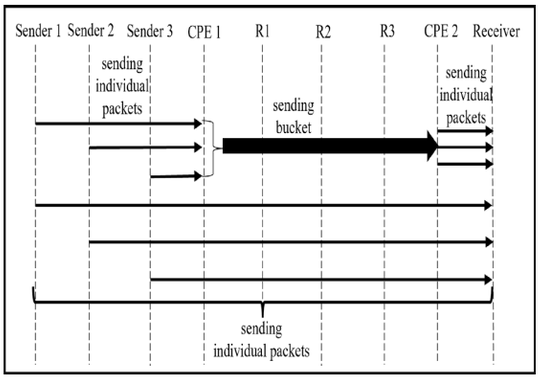

FIGURE 7. Forwarding sequence from senders to receiver

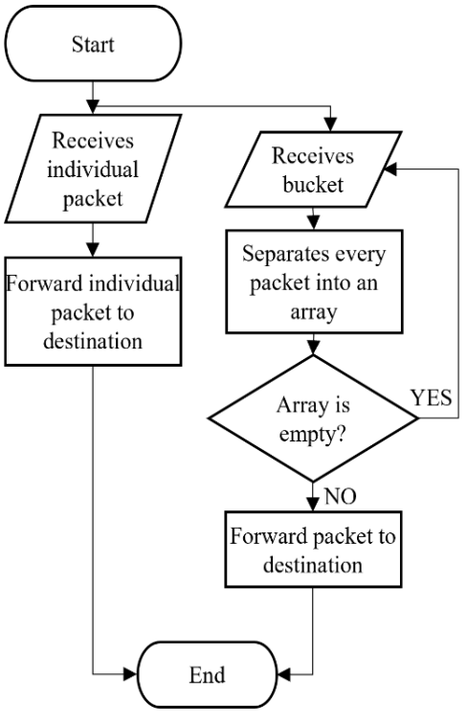

FIGURE 8. Algorithm in CPE 2

Following the implementation of the algorithm, its throughput is evaluated against a standard SD-WAN setup using Graphical Network Simulator 3 (GNS3), which provides access to common network components such as Cisco routers, switches, and OpenFlow switches. Furthermore, OpenFlow switch in GNS3 has built in loss-based congestion control algorithms which are Reno (Ait-Hellal & Altman, 2000) and CUBIC as shown in Figure 10. Reno relies exclusively on packet loss as the trigger for backoff. In these algorithms, loss serves as the binary indicator of congestion. Whenever there are three duplicate ACK packets arrived, the CWND is halved and linear growth resumes. More recent loss-based variants, CUBIC (Bruhn et al. 2023), use a cubic function of the time elapsed since the last reduction to govern window growth. CUBIC's cubic curve allows it to increase its window more aggressively on high-bandwidth, high-delay paths while sharply reduces the sending rate, so it can quickly relieve network queues and avoid further packet drops. These loss-based algorithms are widely used in modern network (Korbi et al. 2024). In this simulation, we let the switch set the algorithm automatically.

The network configuration in GNS3 mirrors that of the proposed design, as illustrated in Figure 2. Data transmission within GNS3 is facilitated using Ostinato as the traffic generator, with data sizes identical to those used in the proposed implementation. A detailed analysis of the results is presented in the subsequent section.

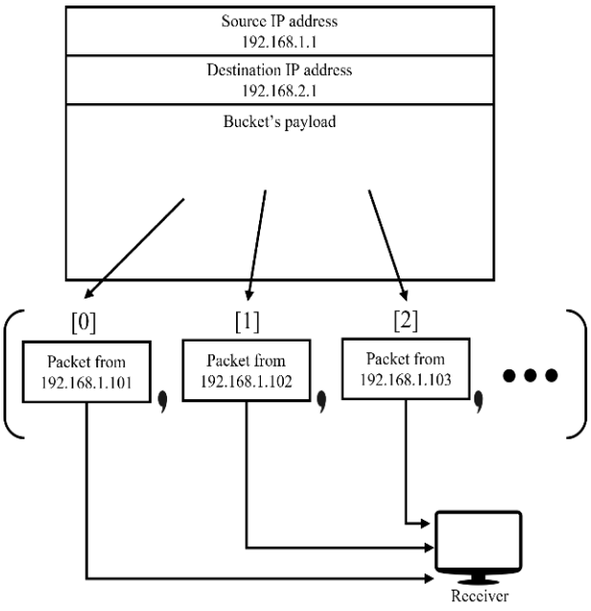

FIGURE 9. Sorted bucket's payload

```
OpenvSwitch-1:/$ sysctl net.ipv4.tcp_available_congestion_control
net.ipv4.tcp_available_congestion_control = reno cubic
```

FIGURE 10. Default congestion control algorithms in GNS3

RESULTS AND DISCUSSION

This section presents the results of the simulation, focusing on a comparative analysis between the proposed algorithm and a standard off-the-shelf congestion control method in GNS3, using throughput as the primary performance metric. The purpose of this comparison is to validate the effectiveness of the proposed algorithm and demonstrate its potential to improve throughput. Throughput is defined as successful received data over time (Moudi & Othman, 2020). In order to obtain the throughput, the end-to-end delay is measured between the two CPEs as shown in Figure 11. This segment aggregates all traffic from the clients and represents the most congested point in the network. The initial throughput is quantified in bits per second (bps). Wireshark is utilized to record the end-to-end delay between two CPEs in Figure 11.

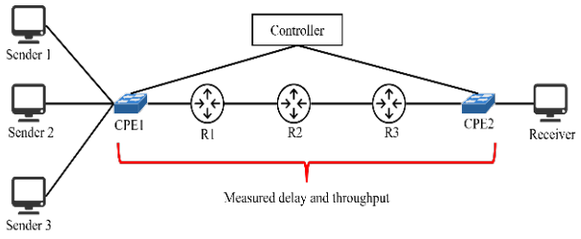

FIGURE 11. Location of measurement

Table 3 shows the comparisons of end-to-end delay between CPE 1 and CPE 2 for proposed algorithm and GNS3. This table shows that the proposed algorithm produces the highest end-to-end delay when transmitting 25 B which is 35.05 s. This is because it takes more time to fill up the bucket due to small data size. Then, the delay values reduce when transmitting 50 B to 200 B. The values vary from 33.8 s to 34.6 s. On the other hand, GNS3 produces higher delay compared to proposed algorithm which varies from 34.08 s to 34.83 s. Overall, the proposed algorithm produces lower end-to-end delay compared to GNS3. Figure 12 illustrates the graphs that reflect the values in Table 3.

TABLE 3. Comparisons of end-to-end delay between proposed algorithm and GNS3

Transmitted data (B)	End-to-end delay (s)	
	GNS3	Proposed algorithm
25	35.72	35.05
50	34.09	33.81
75	34.40	34.33
100	34.35	34.04
125	34.71	34.36
150	34.53	33.88
175	34.68	34.46
200	34.83	34.58

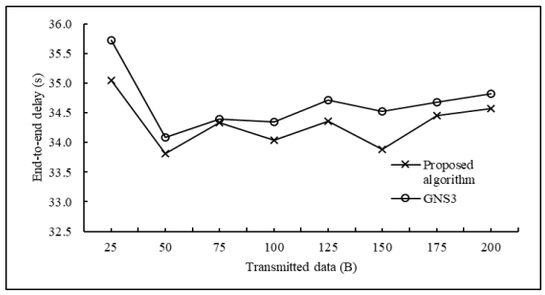

FIGURE 12. Comparisons of end-to-end delay graphs between proposed algorithm and GNS3

Then, the initial throughput, T_i, can be calculated by using Equation (3). In Equation (3), the number of successful transmitted data is 100% because we are using TCP transmissions.

$$T_i\ (bps) = \frac{Successful\ transmitted\ data\ (bits)}{End-to-end\ delay(s)} \quad (3)$$

Tables 4 present comparison of throughputs between the proposed algorithm and the GNS3. The results indicate that throughput increases for both methods as the volume of transmitted data grows. Notably, the proposed algorithm consistently achieves higher throughput which from 5.71 bps at 25 B to 46.71 bps at 200 B. Compared to GNS3, it produces throughput from 5.60 bps at 25 B to 45.94 bps at 200 B. This improvement can be attributed to the relatively small data sizes being transmitted over a high-bandwidth link. Based on Table 4, we can see that the proposed algorithm achieves higher throughput because each router takes lesser time to look up the forwarding

table for the bucket instead of looking up for each individual packet. Figure 13 shows the graphs of throughput comparisons between proposed algorithm and GNS3 proving the performance produced by the proposed algorithm is on par with GNS3's. The readings above the graph reflect the throughput produced by proposed algorithm, while the readings below the graph reflect the throughput produced by GNS3.

TABLE 4. Initial throughput comparisons between proposed algorithm and GNS3

Transmitted data (B)	Initial throughput (bps)	
	GNS3	Proposed algorithm
25	5.60	5.71
50	11.74	11.83
75	17.44	17.48
100	23.29	23.50
125	28.81	29.10
150	34.75	35.42
175	40.37	40.63
200	45.94	46.27

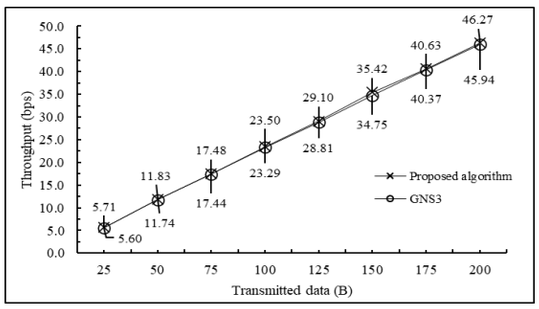

FIGURE 13. Comparisons of throughput graphs between proposed algorithm and GNS3

Then, we scale up the throughput to match the standard of uplink and downlink in today's network. As mentioned in the Introduction section, the uplink and downlink bandwidths are 10 Gbps and 20 Gbps respectively. To get the scaled throughput, we calculate the link utilization, L_i, in a 1 Gbps link by dividing the initial throughput with the link bandwidth as shown in Equation (4) (Nsaif et al. 2024). Link utilization is defined as how much bandwidth is consumed when traffics are transmitted.

$$L_i = \frac{T_i}{1 \times 10^9} \tag{4}$$

Based on Table 5, the proposed algorithm produces better link utilization compared to GNS3. The link utilizations for proposed algorithm increase from 0.57×10^8 at 25 B to 4.63×10^8 at 200 B. This is because the packets are sent in bulk in a bucket resulting individual packets have more space in the link. On the other hand, the link utilizations for GNS3 are from 0.56×10^8 at 25 B to 4.59×10^8 at 200 B.

TABLE 5. Link utilization comparisons between proposed algorithm and GNS3

Transmitted data (B)	Link utilizations ($\times 10^8$)	
	GNS3	Proposed algorithm
25	0.56	0.57
50	1.17	1.18
75	1.74	1.75
100	2.33	2.35
125	2.88	2.91
150	3.48	3.54
175	4.04	4.06
200	4.59	4.63

Then, to get the scaled throughput, the link utilization is multiplied with 10 Gbps link and 30 because the packets are sent 30 times as shown in Equation (5). In Equation (5), 10 Gbps is replaced with 20 Gbps to get the scaled throughput for downlink transmissions.

$$Scaled\ throughput = L_i \times 30 \times 10\ Gbps \tag{5}$$

After scaling, the throughputs show more noticeable improvements between proposed algorithm and GNS3 compared to the initial throughput comparison. Even the link utilization is small, Table 6 shows the proposed algorithm can improve thousands of bits per second. In this simulation, the scaled of throughputs of the proposed algorithm are from 1,712 kbps to 13,882 kbps for uplink transmissions. Compared to GNS3, the scaled uplink throughputs are from 1,680 kbps to 13,781 kbps. The throughput differences between proposed algorithm and GNS3 are from 10 kbps to 200 kbps which can be translate to about 0.19% to 1.92% of improvements.

On the other hand, Table 7 shows the scaled throughputs of proposed algorithm when in 20 Gbps link are from 3,423 kbps to 27,764 kbps. Meanwhile, the scaled downlink throughputs for GNS3 are from 3,359 kbps to 27,562 kbps. The differences of throughputs between proposed algorithm and GNS3 are from 20 kbps to 400 kbps which are also about 0.19% to 1.92% of improvements. Hence, we can predict that as link capacity increases, the improvement of throughput will be higher. This proves that the proposed algorithm performs better in data-intensive

network. Figure 14 and Figure 15 shows the comparisons of scaled throughputs for uplink and downlink transmission respectively. Similar to Figure 13, the scaled throughputs of proposed algorithm are on par with GNS3. The readings above the graphs reflect the scaled throughput produced by proposed algorithm, while the readings below the graphs reflect the scaled throughput produced by GNS3. Furthermore, Figure 16 shows the improvements of uplink and downlink transmissions. In the next section, we conclude this paper.

TABLE 6. Scaled throughput comparisons for uplink transmissions

Transmitted data (B)	Uplink transmissions throughput in 10 Gbps link (kbps)		Uplink transmissions improvements (%)
	GNS3	Proposed algorithm	
25	1,680	1,712	1.91
50	3,521	3,549	0.80
75	5,233	5,243	0.19
100	6,987	7,050	0.89
125	8,642	8,731	1.03
150	10,425	10,625	1.92
175	12,111	12,188	0.64
200	13,781	13,882	0.73

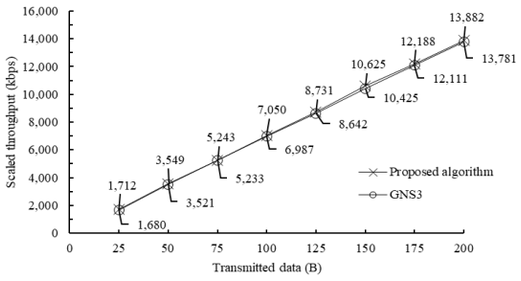

FIGURE 14. Comparisons of uplink throughput graphs between proposed algorithm and GNS3

TABLE 7. Scaled throughput for downlink transmissions

Transmitted data (B)	Downlink transmissions throughput in 20 Gbps link (kbps)		Downlink transmissions improvements (%)
	GNS3	Proposed algorithm	
25	3,359	3,423	1.91
50	7,041	7,098	0.80
75	10,466	10,486	0.19
100	13,975	14,099	0.89
125	17,284	17,461	1.03

continue...

...cont.

150	20,851	21,251	1.92
175	24,221	24,376	0.64
200	27,562	27,764	0.73

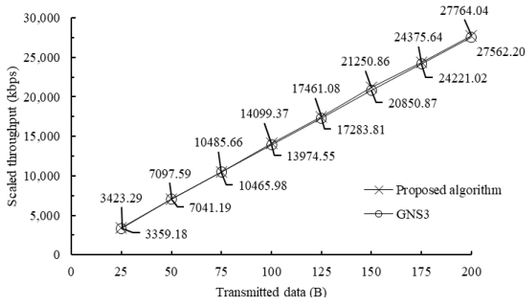

FIGURE 15. Comparisons of downlink throughput graphs between proposed algorithm and GNS3

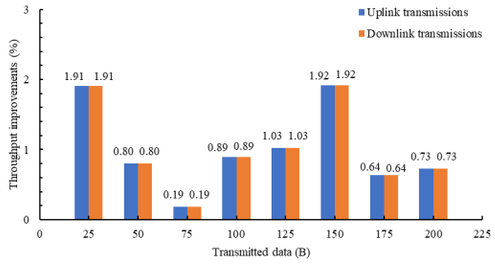

FIGURE 16. Throughput improvements for uplink and downlink transmissions

CONCLUSION

WAN is fundamental in enabling communications between diverse platforms and end devices. However, the high operational costs associated with WAN infrastructure have prompted the emergence of SDN as a solution. SDN addresses this challenge by decoupling the control plane from the data plane, while also introducing programmability into network nodes. Hence, enhancing flexibility and adaptability. The integration of WAN and SDN gives rise to SD-WAN, which offer improved QoS at a reduced cost. As technology continues to advance, the volume of network traffic also increases significantly. This causes the network to be congested, but at the same time, the traffic needs to be delivered fast. Therefore, congestion control mechanisms emerged to solve this issue especially for TCP transmission. Over the recent years, researchers have proposed several state-of-the-art congestion control mechanisms. However, existing mechanisms typically respond only after congestion has already occurred. In order to address this limitation, this paper proposes a proactive congestion control approach based on the bucket sort algorithm. The

algorithm aggregates packets sharing the same destination network IP address into a single bucket and transmits them as one unit. The performance in terms of throughput of the proposed method is then evaluated against a standard loss-based congestion control approach in GNS3. The result shows that the proposed algorithm produces better throughput which is up to 46.27 bps compared to GNS3's which is only up to 45.94 bps. Furthermore, the proposed algorithm has better link utilization which is up to 4.63×10^8 compared to GNS3's which is up to 4.59×10^8. Better link utilizations result in better throughputs for uplink and downlink transmissions. By using the values of link utilizations, the throughputs can be scaled up to the current communication standards which are 10 Gbps uplink transmissions and 20 Gbps for downlink transmissions. The maximum scaled throughput for uplink transmissions produced proposed algorithm is 13,882 kbps, while its maximum downlink throughput is 27,764 kbps. Compared to GNS3, its maximum uplink throughput is 13,781 kbps, while its maximum downlink throughput is 27,562 kbps. Based on these results, it can be concluded that the proposed algorithm able to perform better in high volume of traffics compared to GNS3.

ACKNOWLEDGEMENT

The authors would like to thank Universiti Tenaga Nasional (UNITEN) for the financial support through the BOLD Grant 2024 (Project Code: J510051040)

DECLARATION OF COMPETING INTEREST

None.

REFERENCES

Abane, A., Cubeddu, M., Mai, V. S., & Battou, A. 2025. Entanglement Routing in Quantum Networks: A Comprehensive Survey. *IEEE Transactions on Quantum Engineering* 6: 1-39

Ait□Hellal, O., & Altman, E. 2000. Analysis of TCP Vegas and TCP Reno. *Telecommunication systems* 15(3): 381-404

Alotaibi, H. S., Gregory, M. A., & Li, S. 2022. Multidomain SDN□Based Gateways and Border Gateway Protocol. *Journal of Computer Networks and Communications* 2022(1): 1-23

Alwasel, K., Jha, D. N., Hernandez, E., Puthal, D., Barika, M., Varghese, B., Garg, S. K., James, P., Zomaya, A., & Morgan, G. 2020. IoTSiM-SDWAN: A Simulation Framework for Interconnecting Distributed Datacenters Over Software-Defined Wide Area Network (SD-WAN). *Journal of Parallel and Distributed Computing* 143: 17-35

Andújar, F. J., Sánchez de la Rosa, M., Escudero-Sahuquillo, J., & Sánchez, J. L. 2023. Extending the VEF traces framework to model data center network workloads. *The Journal of Supercomputing* 79(1): 814-831

Bruhn, P., Kuehlewind, M., & Muehleisen, M. 2023. Performance and improvements of TCP CUBIC in low-delay cellular networks. *Computer Networks* 224: 109609

Chefrour, D. 2021. One-way delay measurement from traditional networks to sdn: A survey. *ACM Computing Surveys (CSUR)* 54(7): 1-35

George, A. S., George, A. H., & Baskar, T. 2023. SD-WAN Security Threats, Bandwidth Issues, SLA, and Flaws: An In-Depth Analysis of FTTH, 4G, 5G, and Broadband Technologies. *Partners Universal International Innovation Journal* 1(3): 1-37

Hussain, I., & Bashir, J. 2022. Dynamic MTU: A Smaller Path MTU Size Technique to Reduce Packet Drops in IPv6. *Journal of King Saud University-Computer and Information Sciences* 34(9): 7070-7088

Jowkarishasaltaneh, F., & But, J. 2022. An Analysis of MPTCP Congestion Control. *Telecom* 3(4): 581-609

Korbi, Y., Zhani, M. F., & Kaippallimalil, J. 2024. Congestion Control in Wi-Fi Networks-State of the Art, Performance Evaluation, and Key Research Directions. *IEEE Access* 12: 94972 - 94989

Leivadeas, A., & Falkner, M. 2022. A survey on intent-based networking. *IEEE Communications Surveys & Tutorials* 25(1): 625-655

Lorincz, J., Klarin, Z., & Ožegović, J. 2021. A comprehensive overview of TCP congestion control in 5G networks: Research challenges and future perspectives. *Sensors* 21(13): 4510

Mahafzah, B. A. 2024. Parallel bucket sort algorithm on optical chained-cubic tree interconnection network. *Jordanian Journal of Computers and Information Technology (JJCIT)* 10(3): 319-334

Moudi, M., & Othman, M. 2020. On the relation between network throughput and delay curves. *Automatika* 61(3): 415-424

Nguyen, M., & Debroy, S. 2022. Moving Target Defense□Based Denial□of□Service Mitigation in Cloud Environments: A Survey. *Security and Communication Networks* 2022(1): 1-24

Nsaif, M., Kovásznai, G., Malik, A., & de Fréin, R. 2024. SM-FPLF: Link-state prediction for software-defined DCN power optimization. *IEEE Access* 12: 79496-79518

Obiodu, E., & Sastry, N. 2020. From ATM to MPLS and QCI: The Evolution of Differentiated QoS Standards and Implications for 5G Network Slicing. *IEEE Communications Standards Magazine* 4(2): 14-21

Ouamri, M. A., Alharbi, T., Singh, D., & Sylia, Z. 2025. A comprehensive survey on software-defined wide area network (SD-WAN): principles, opportunities and future challenges. *The Journal of Supercomputing* 81(1): 1-47

Quang, P. T. A., Martin, S., Leguay, J., Gong, X., & Huiying, X. 2022. Intent-based routing policy optimization in SD-WAN. *ICC 2022-IEEE International Conference on Communications* 4914-4919

Ridwan, M. A., Radzi, N. A. M., Wan Ahmad, W. S. H. M., Abdullah, F., Jamaludin, M. Z., & Zakaria, M. N. 2020. Recent trends in MPLS networks: technologies, applications and challenges. *IET Communications* 14(2): 177-185

Sahu, A., Venkatraman, V., & Macwan, R. 2023. Reinforcement Learning Environment for Cyber-Resilient Power Distribution System. *IEEE Access* 11: 127216-127228

Shojaee, M., Neves, M., & Haque, I. 2020. SafeGuard: Congestion and memory-aware failure recovery in SD-WAN. *2020 16th International Conference on Network and Service Management (CNSM)* 1-7

Song, C., & Li, H. 2023. Improvement of counting sorting algorithm. *Journal of Computer and Communications* 11(10): 12-22

Tan, Y., Veeraraghavan, M., Lee, H., Emmerson, S., & Davidson, J. W. 2022. High-performance reliable network-multicast over a trial deployment. *Cluster Computing* 25(4): 2931-2952

Tong, V., Souihi, S., Tran, H. A., & Mellouk, A. 2024. Troubleshooting solution for traffic congestion control. *Journal of Network and Computer Applications* 229: 1-24

Troia, S., Mazzara, M., Savi, M., Zorello, L. M. M., & Maier, G. 2022. Resilience of Delay-sensitive Services with Transport-layer Monitoring in SD-WAN. *IEEE Transactions on Network and Service Management* 19(3): 2652-2663

Troia, S., Sapienza, F., Varé, L., & Maier, G. 2020. On Deep Reinforcement Learning for Traffic Engineering in SD-WAN. *IEEE Journal on Selected Areas in Communications* 39(7): 2198-2212

Watanabe, M., & Hasegawa, G. 2025. Design, discussion, and its potential performance evaluation of in-network congestion control. *IEICE Transactions on Information and Systems* E108.D(11): 1302-1314

Yang, D., & Tsai, W.-T. 2024. SDN-Based Congestion Control and Bandwidth Allocation Scheme in 5G Networks. *Sensors* 24(3): 749

Zebari, G. M., Zebari, D. A., & Al-zebari, A. 2021. Fundamentals of 5G cellular networks: A review. *Journal of Information Technology and Informatics* 1(1): 1-5

Zhang, Y., Tourrilhes, J., Zhang, Z.-L., & Sharma, P. 2021. Improving SD-WAN Resilience: From Vertical Handoff to WAN-Aware MPTCP. *IEEE Transactions on Network and Service Management* 18(1): 347-361