*Journal of Computer Science*

Research Article

# MFQ: Network Bandwidth Utilization among Data Centers on Software Defined Network Using Multilevel-Feedback Queue

**[1]R Shanthy, [2]K Sutha and [3]J Emerson Raja**

[1]*Department of Computer Science, Sona College of Arts and Science, Salem, Tamilnadu, India*
[2]*Department of Computer Science and Applications, Faculty of Science and Humanities, SRM Institute of Science and Technology, Ramapuram Campus, Chennai, India*
[3]*Faculty of Engineering and Technology, Multimedia University, Melaka, Malaysia*

*Article history*
Received: 30-07-2024
Revised: 26-10-2024
Accepted: 06-11-2024

Corresponding Author:
J Emerson Raja
Faculty of Engineering and Technology, Multimedia University, Melaka, Malaysia
Email: emerson.raja@mmu.edu.my

**Abstract:** The growing complexity of network traffic patterns and congestion within data centers necessitate innovation in traditional network infrastructure. While advancements in storage, computing, and applications have progressed rapidly, networking has remained relatively stagnant, limiting the overall performance of data centers. Software-Defined Networking (SDN) emerges as a powerful solution by decoupling the control plane from the data plane, offering enhanced flexibility and efficiency in network management. This decoupling allows for centralized control and dynamic configuration, making SDN well-suited for modern, high-demand environments. As network architectures evolve, SDN is expected to enable more adaptive and context-sensitive routing protocols that can adjust to real-time conditions. However, challenges in optimizing resource utilization across data centers persist. This paper proposes a novel approach utilizing a multilevel-feedback queue mechanism to enable real-time traffic processing and dynamic scheduling, improving bandwidth utilization. Simulation results demonstrate the effectiveness of this method in optimizing traffic scheduling, highlighting its potential for addressing congestion issues and enhancing network performance in data centers.

**Keywords:** Bandwidth, Congestion Control, Data Center, Dynamic Scheduling, Software Defined Network

## Introduction

Research data is growing at a very high rate due to the development in big data analysis, the amount of data from various sources such as social media, blogs, and sensors from the Internet of Things, and cloud computing data (Laizhong Cui & Yan, 2016). Low utilization ratio, congestion problems, service latency, and even DDOS attacks may be due to the heavy data transmission impact (Lan *et al.*, 2016). All data centers are always linked to each other through wide-area networks (Ghaffarinejad & Syrotiuk, 2014). When data center networks employ traditional routing protocols, flows are compelled to forego the shortest way in order to be routed and forwarded. This may be the shortest path connection when all links are under low load and some fresh flows are still vying for it. The best but most expensive solution is to reshape and update the network. However, traditional methods of configuring network systems, monitoring, and optimizing network performance, identifying and resolving network problems, and planning network expansion will become nearly impossible and ineffective due to the complexity, scalability, and heterogeneity of current networks (Xia *et al.*, 2015). One of the most popular types of computer networking is Software Defined Networking (SDN). SDN's central concept is to decouple the control plane from the data plane through software programming to achieve flexible network management, effective network activity, and low-cost maintenance (Sezer *et al.*, 2013). It is receiving considerable attention from academic researchers, industry researchers, network operators, and some large and medium-sized networking enterprises (Nunes *et al.*, 2014). SDN is seen as a promising approach for architecting our conventional network (Lin *et al.*, 2015). Traditional SDN's basic concept is to decouple the control plane from the data plane to accomplish flexible network management, effective network activity, and low-cost maintenance through software programming (Sezer *et al.*, 2013). In particular, network tools merely perform packet forwarding based on SDN controller principles (McKeown *et al.*, 2008). The SDN administrator will oversee the underlying network architecture in the control plane and give the

SCIENCE Publications

© 2025 R Shanthy, K Sutha and J Emerson Raja. This open-access article is distributed under a Creative Commons Attribution (CC-BY) 4.0 license.

equipment plane a flexible and efficient environment in which to implement different network services. Under this modern network model, creative approaches are also applied rapidly and reliably inside the kind of software to realize unique and scalable systems and deployed under real-time traffic networks. Nonetheless, this paradigm enables the logically centralized control and management of network devices within the data plane in conjunction with a global and simultaneous network view and real-time network data. Compared to traditional networks, SDN makes it much simpler to develop and implement applications (Kim & Feamster, 2013).

### *Related Work*

This segment looks at the data center and theoretical base traffic engineering survey. The problems include data centers based on conventional and SDN-based networks. First, we highlight the drawbacks of conventional network protocols used in existing data centers and the emergence of distributed data centers. These limitations drive the demand for our proposed work.

### *Data Center Based on Traditional Network*

In the conventional model, traffic primarily flows between the server and the client, while data centers handle only a small portion of the traffic. The majority of network activity is directed towards end-user devices and data centers serve more as storage and processing hubs, with limited involvement in the overall data exchange. This results in an inefficient allocation of network resources, as data centers are not optimized for the growing demand of modern workloads (Greenberg *et al.*, 2008). new network technologies and developments such as the Vehicle Web (Cheng *et al.*, 2015), big data, and cloud computing have emerged to address the challenges of managing vast amounts of data and large-scale distributed data centers. These innovations have led to the rapid increase in east-west traffic between servers, which is primarily driven by the need for efficient data processing and storage. For example, systems like the Google File System (GFS) (Ghemawat *et al.*, 2003) and the Hadoop Distributed File System (HDFS) have been designed to handle this surge in traffic, facilitating seamless data access and distribution across multiple servers within a data center. Additionally, the Google MapReduce framework (Shvachko *et al.*, 2010) has been developed to process large-scale data sets in a distributed manner, further optimizing data flow and computation in these environments (Dean & Ghemawat, 2008). However, congestion occurs frequently due to the use of conventional routing protocols within the shortest path connection. And, due to the delay and low latency, it can further reduce the consistency of network output. Traffic routing and congestion management are two key factors in maintaining network power and increasing network performance. There are some congenital flaws in conventional networks. The main drawbacks are as follows: First, conventional networks lack a coordinative global optimization approach. Each node applies traffic control functions independently, limiting the ability to achieve global optimization. As a result, these systems can only reach optimal conditions locally within each node, failing to adapt dynamically to broader network conditions. Moreover, conventional networks do not support complex, self-adaptive changes that could automatically adjust to varying traffic patterns or network disruptions. In addition, these networks struggle to gain efficient and accurate control over individual network units. The configurations of computers in these networks are often highly complex, making it difficult to identify and resolve configuration-induced network errors due to the intricacies of their command structures. To address these challenges, we leverage Software-Defined Networking (SDN) to streamline and resolve these issues within data centers. By centralizing control, SDN enables more efficient handling of network traffic and congestion management, providing a dynamic and scalable solution to optimize network performance across distributed data center environments.

### *SDN - Data Center*

More visibility and fine-grained control over the entire network are possible with SDN designs. During a position, the SDN controller is prepared to configure data plane infrastructure forwarding devices to track and fine-tune network packets as they flow through them. Therefore, we shall require the periodic gathering of specific statistics using the SDN controller. Additionally, we may notify up-level apps of a change on a real-time network and obtain a centralized picture of the network state for SDN applications through open APIs (Ali *et al.*, 2015).

Data centers typically have thousands of fast connections, such as Ethernet Using NOX, the first SDN controller, to successfully improve the addressing and routing of the current VL2 and Portland data center networks, (Tavakoli *et al.*, 2019) initially deployed SDN to data center networks using traditional packet capture techniques. HyperFlow, a substitute control scheme provided to OpenFlow, has been proposed by Tootoonchian & Ganjali (2010). While maintaining the benefit of centralized management, several controllers work together to compensate for the limited scalability of a single controller. Network growth and straightforward device management are made possible by the collaboration of dispersed controllers (Yu *et al.*, 2014; Zhang *et al.*, 2017). Hindman and associates (Koponen *et al.*, 2010; Benson *et al.*, 2011; Hindman *et al.*, 2011) bensons, Mesos presented a platform for fine-grained resource sharing in data centers. MicroTE is a system that adapts to traffic variations by leveraging the short-term and partial predictability of the traffic matrix. They implement MicroTE within the OpenFlow framework

with minor modifications to the end host. The system performs close to the optimal solution and imposes minimal overhead on the network making it appropriate for current and future data centers. Mahout was suggested by Curtis *et al.* (2011) to manage elephant flows or huge flows when large flows were observed in socket buffers. Congestion and equipment failure are common because of traffic congestion and consequent equipment failure. There is frequently a combination of large and small flows in data center networks. Kanagavelu *et al.* (2012) proposed a flow-based, edge-to-edge rerouting system. When congestion developed, it focused on diverting the heavy flows to other links. This mechanism is explained by the fact that varying such brief tiny flows between links would also result in higher overhead and latency. A layer between the so-called VeriFlow units and a software-defined network controller was supplied by Khurshid *et al.* (2012). It is a network monitoring tool for identifying unusual network activity and flawed SDN application rules. In order to lessen the performance deterioration of intensively used links, (Tso & Pezaros, 2013) proposed the Baatdaat flow scheduling technique leveraging the spare data center network capacity. It will secure the intricate timetable in real time, preventing congestion brought on by instant flows. In order to allow the route to be dynamically adjusted in accordance with the overall network view, Li *et al.* (2014) proposed a traffic scheduling system that included the Fuzzy Synthetic Assessment Process. In conclusion, the majority of current active techniques for traffic engineering in SDN-supported data center networks have a number of possibilities.

Our proposed scheme is an initiative that first applies the multi-level feedback queue mechanism to traffic scheduling and SDN-supported congestion management across data centers and can also provide a new congestion solution caused by the persistent use of malicious flows (Shukla *et al.*, 2023) Monitoring traffic flow ensures that anomalies are accurately identified. The effectiveness of cyber-attacks in SDN is enhanced by the suggested method. Remarkable attack detection performance is achieved by the suggested model in the case of Distributed Denial-of-Service (DDoS) attacks, as network forwarding performance degradation is prevented. Traffic flows are matched by users in ways that increase granularity while the SDN data plane is proactively protected from overloading the performance of cyber-attack detection (Alhilali & Montazerolghaem, 2023).

For effective domain-wide traffic routing and administration, SDN divides the control and data planes. Traffic distribution issues with SDN, including load imbalance, can have a detrimental impact on network performance. In order to increase SDN efficacy, developers have created a variety of SDN load-balancing solutions. Because the science of Artificial Intelligence (AI) is expanding quickly, some AI techniques are being incorporated into SDN to enhance network performance and resource usage. The first step is to examine the SDN architecture and look into the load balancing issue. Second, classifying AI-based load balancing techniques and carefully evaluating these mechanisms from a variety of angles, including the problem addressed, the algorithm or methodology used, and their advantages and disadvantages.

Thirdly, providing an overview of the measures used to gauge how effective these methods are Huang *et al.* (2021). In order to optimize QoS in a hybrid SDN, the author suggests a nearly ideal traffic control technique. To enhance the results of optimization, an SDN migration sequence is first investigated in order to maximize controllable traffic. The multi-splitable routing issue in the hybrid SDN is then handled using a Deep Reinforcement Learning (DRL) method. The OpenFlow group bucket limits are defined in order to apply the flow split ratio method. Lastly, we use open-source traffic datasets to assess the suggested approach.

## Materials and Methods

### *The Design and Implementation of MFQ*

Using the flexibility and agility of SDN, the MFQ architecture addresses traffic routing and congestion control across numerous data centers. The data centers are frequently connected by extensive area networks. The network traffic has an uneven flow distribution and is not always consistent. During peak hours, network traffic may more than double compared to a typical load connection. However, the traditional routing protocols route and forward flow without pushing off flows to balance the connection load according to the shortest path algorithm. We are forced to upgrade to large-scale routing equipment and buy 2–3× bandwidth in order to meet the bandwidth requirements during peak hours. Network service, maintenance, and administration Charges are consequently sharply raised, which frequently results in bandwidth waste during regular business hours. In this study, we propose MFQ, a dynamic scheduling scheme for traffic and congestion management with a multilevel feedback queue system that can to some degree resolve the above-mentioned problems. The following sectors illustrate in depth the layout and algorithm of the scheme and its method of implementation.

### *Scheme Model*

Data center networking served as the foundation for our deliberate and planned scheme approach. We provide a quick summary of the Scheme model's elements in Figure (1). (I) The data center servers. Three distinct data centers are PC1, PC2, and PC3. Traffic flows from PC1-3 are prioritized over those from PC2-3. (ii) Two distinct paths (S1, S4, S5) and the second shortest path (S1, S2, S3, S4) are used to produce the five OpenFlow switches.

(iii) SDN-equipped controller. Floodlight, the centralized system, can be used to track switch actions by updating the flow table in the switches.

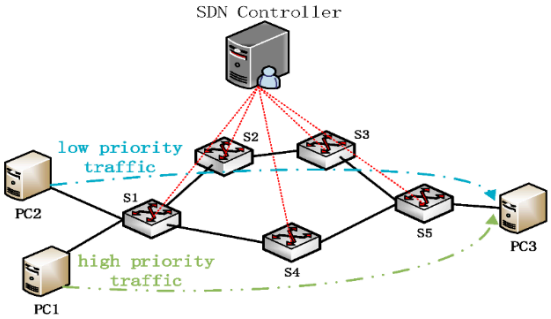


**Fig. 1:** Scheme model

### *Dynamic Traffic Scheduling*

Several data centers classify traffic flows into different categories. While other high-quality criteria data flows had a higher priority flow, we established a lower priority flow for data replication. Depending on the condition of the real-time link, the lower priority data flow might select a path, while the higher priority data flows follow the shortest way.

#### *The Shortest Paths are Divided into the Following Four Cases*

Case 1: The time frame during which a low-priority flow arrives over the network. Here, the options are still expanding. First, it will choose the way directly if the bandwidth remains on the shortest path in real-time (S1, S4, S5). A high-priority flow will be forced to use the second shorter path (S1, S2, S3, S5) if it takes up the entire shortest path (S1, S4, S5). Last but not least, if there is no bandwidth available for any of the paths, this flow of multilevel feedback queues will join the congestion control.

Case 2: A time when the link saw a high-priority flow. There are a number of options available to us in this situation. First, it will choose the way directly if the bandwidth stays on the shortest path (S1, S4, S5). A high-priority flow would be forced to use the second, shorter way (S1, S2, S3, S5) if it took up the entire shortest path (S1, S4, S5). Since there isn't enough bandwidth for all paths, the final option is that this flow will combine multilevel feedback queues with congestion control.

Case 3: The time frame on the connection when a high-priority flow transmits. Due to the high-priority data flow, we are not taking any action.

Case 4: The time frame on the connection when a high-priority flow transmits. We will discover from instance 1 that the low-priority flow can choose from either of the two paths that adjust to data fluctuations. As a result, the shortest path link will be vacated and set aside for the second shortest path connection if there is no bandwidth left on it when a high-priority replacement flow arrives. Otherwise, the shortest path is given to the newly arrived flow if the new data flow has an equally low priority and no further transmission of low priority flow is necessary. Based on the monitoring of connection statuses, we might implement dynamic traffic scheduling to maximize the use of bandwidth resources.

### *Congestion Control*

As part of MFQ, complex traffic management has an obvious impact on the company and flow diversion. Due to the long-term occupation of limited connection capital and the burst of network prompt traffic, congestion may also occur in the shortest route path. This congestion control algorithm provides a queuing service for congested flow and also settles the vicious prolonged occupation problem. We define two multilevel Feedback queues to store the flows that await scheduling. One is a low priority queue for multi-level reviews and the other is of high priority. Then we describe three sub-queues in each feedback queue and give top priority to flow waiting queue 1, second highest priority to flow waiting queue 2, and lowest priority to flow waiting queue 3. As shown in Figure (2), after scheduling, the transmission time of these waiting queues for flows is different. Sub-queue 1 has time t, sub-queue 2 has time 2 t which is twice the sub-queue 1 time, and sub-queue 3 has time 3 t which is triple the sub-queue 1 time.

**Algorithm 1**: The Algorithm of Multilevel-Feedback queue

**Input:** G: Topology of data center network;
$S_{active}$: a set of active flows;
$S_{suspend}$: a set of suspended flows;
$S_{new}$: a new flow;
$S_{first}$: a flow from the first of the $S_{suspend}$.

**Output:** {<e.state, e.path>}: scheduling state and path selection of each flow in G. When a new flow arrives

**if** (e.path = IDLEPATH) **then**
  $S_{active} \leftarrow S_{active} + S_{new}$;
**end**
**else** $S_{suspend} \leftarrow S_{suspend} + S_{new}$;
**while** ($S_{suspend} \neq \varnothing$) **do**
  **if** (e.path = IDLEPATH) **then**
    **if** ($S_{suspend1} \neq \varnothing$) **then**
      Select the flow at the first of $S_{suspend1}$;
      $S_{suspend} \leftarrow S_{suspend} - S_{first}$;
      $S_{active} \leftarrow S_{active} + S_{first}$;
      Transmission time of the first is *t*;
    **end**
    **else if** ($S_{suspend2} \neq \varnothing$) **then**
      Select the flow at the first of $S_{suspend2}$;
      $S_{suspend2} \leftarrow S_{suspend2} - S_{first}$;
      $S_{active} \leftarrow S_{active} + S_{first}$;
      Transmission time of the $S_{first}$ is 2*t*;
    **end**
    **else**
      Select the flow at the first of $S_{suspend3}$;
      $S_{suspend3} \leftarrow S_{suspend3} - S_{first}$;
      $S_{active} \leftarrow S_{active} + S_{first}$;
      Transmission time of the $S_{first}$ is 3*t*;
    **end**
**return** {<e.state, e.path>};

**DOI: 10.3844/jcssp.2025.1576.1585**

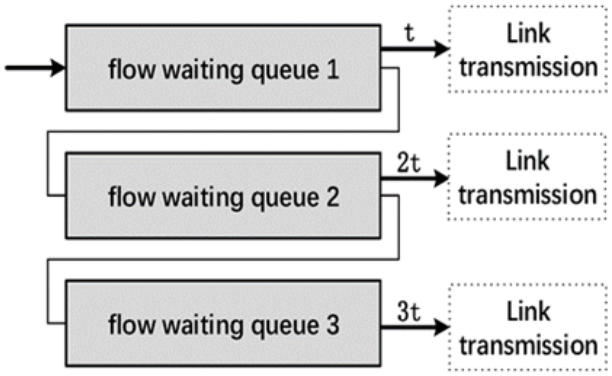


**Fig. 2:** Multilevel-feedback queue

## Results

It illustrates the integration of the open-source Floodlight SDN controller, which operates within an Eclipse environment on the Ubuntu framework**,** as part of our simulation setup. In this configuration, the Floodlight controller serves as the backbone of our Software Defined Networking (SDN) infrastructure, providing centralized control over the network. The MFQ scheme is implemented as an extension on top of the Floodlight controller, allowing us to manage network traffic efficiently using the principles of Multi-Flow Queueing.

In our system, we constructed a virtual switch and set up an SDN environment that enables dynamic network management and control. The virtual switch is responsible for forwarding traffic between the network nodes and the SDN controller oversees the routing of traffic based on the decisions made by the MFQ scheme. This structure ensures that the network is responsive and adaptable to changes in traffic flow, enabling better congestion management and traffic scheduling.

Two key modules are integrated into the system to track and monitor network performance. The first module records critical connection information, including details such as bandwidth usage and congestion levels across various network links. This real-time data is crucial for assessing network health and performance. The second module tracks packet-inandpacket-out messages within the simulation protocol, capturing data related to incoming and outgoing traffic at the network nodes. These messages provide detailed insights into the flow of packets across the network, allowing for granular visibility into how the system is operating. The combination of these modules enables real-time detectionandtraffic control**.** When new data arrives or when a flow needs to be scheduled, the system can instantly detect the need for adjustments and take action. This process ensures that congestion is minimized and the available bandwidth is utilized efficiently. By leveraging the capabilities of the Floodlight controller, our MFQ-based system can dynamically adapt to traffic changes, prioritize flows, and optimize resource allocation in real time, improving overall network performance.

Figure (3) provides a visual representation of the simulation setup, where we modeled two servers, h1, and h2, along with five data centers (c1, c2, c3, c4, and c5) as a unified data center network. In this setup, each server and data center was assigned unique virtual IP addresses and MAC addresses to simulate realistic networking environments. These identifiers helped us accurately track and record the data flows among the data centers during the simulation.

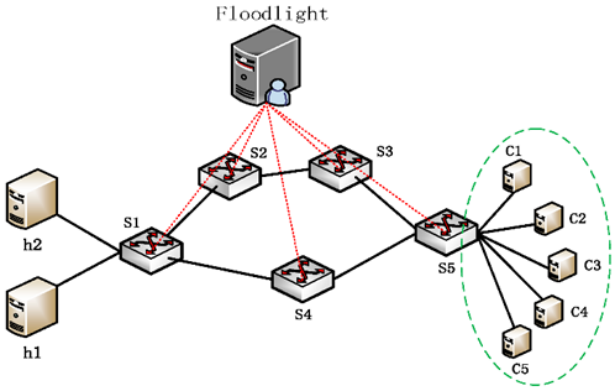


**Fig. 3:** Experimental topology

### *Real-Time Bandwidth-Comparison*

In the system model, we defined two key paths: (s1, s4, s5), which was chosen as the shortest path, and (s1, s2, s3, s4), which was considered the second shortest path. These paths were crucial for evaluating the efficiency and performance of the Multi-Flow Queueing (MFQ) scheme in managing traffic flows within the network. The use of these specific paths allowed us to assess how well MFQ optimizes traffic across both primary and alternative routes. To verify the effectiveness of the MFQ scheme, we conducted two distinct tests. These tests were designed to evaluate how the MFQ mechanism performs in optimizing traffic scheduling and improving network resource utilization under different conditions. By simulating traffic flows along the shortest and second shortest paths, we aimed to observe how well MFQ manages congestion, minimizes packet loss, and maintains high throughput and low latency, even under varying network conditions.

Analysis 1: In this analysis, we configured both the shortest and second shortest paths with a bandwidth of 4 Mbps each. We used the iperf method to simulate four separate traffic flows between the following pairs of nodes: h1 and c1, c2 and h1, as well as h2 and c3 and h4 and c2. Each individual link in the network was set to a bandwidth of 2 Mbps. In addition to setting up the simulation, we evaluated the performance of the conventional network using the Multi-Flow Queueing (MFQ) technique. The data collected during the simulation, including performance metrics for both the

**DOI: 10.3844/jcssp.2025.1576.1585**

MFQ and conventional network, are displayed in Table (1).

To further assess network performance, we calculated the overall usage of the links in both network configurations, which is shown in Figure (4). In terms of link usage, the average utilization of the MFQ-based scheme remained consistently high at 97%, demonstrating its superior efficiency. In contrast, the conventional network's average usage began at a significantly lower rate of just 73%. This marked difference in performance highlights the substantial improvements in link usage when employing the MFQ approach, illustrating its ability to better manage traffic and optimize network resources.

**Table 1:** Real-Time Bandwidth-Comparison

| Bps-Time | Traditional Network | | | | MFQ | | | |
|---|---|---|---|---|---|---|---|---|
| | h1-c1 Transfer/ Bandwidth | h1-c2 Transfer/ Bandwidth | h2-c3 Transfer/ Bandwidth | h2-c4 Transfer/ Bandwidth | h1-c1 Transfer/ Bandwidth | h1-c2 Transfer/ Bandwidth | h2-c3 Transfer/ Bandwidth | h2-c4 Transfer/ Bandwidth |
| 0-1 s | 106 k | 1.87M | 1.91M | 2.53k | 1.94M | 1.94M | 1.94M | 1.95M |
| 1-2 s | 11.8 k | 1.86M | 1.87M | 70.4k | 1.94M | 1.95M | 1.95M | 1.94M |
| 2-3 s | 11. 8k | 1.89M | 1.87M | 36.6k | 1.95M | 1.94M | 1.93M | 1.89M |
| 3-4 s | 153 k | 1.92M | 1.87M | 10.8k | 1.94M | 1.95M | 1.95M | 1.93M |
| 4-5 s | 35.3k | 1.80M | 1.87M | 176k | 1.94M | 1.98M | 1.94M | 1.94M |
| 5-6s | 82.3 k | 1.94M | 1.87M | 80.4k | 1.93M | 1.94M | 1.96M | 1.95M |
| 6-7 s | 35.3 k | 1.95M | 1.87M | 22.5k | 1.95M | 1.95M | 1.97M | 1.94M |
| 7-8 s | 23.5 k | 1.93M | 1.87M | 57.5k | 1.94M | 1.94M | 1.98M | 1.94M |
| 8-9 s | 11.8 k | 1.92M | 1.87M | 22.5k | 1.97M | 1.96M | 1.95M | 1.95M |
| 9-10 s | 23.5 k | 1.91 | 1.87M | 11.9k | 1.87M | 1.87M | 1.95M | 1.94M |

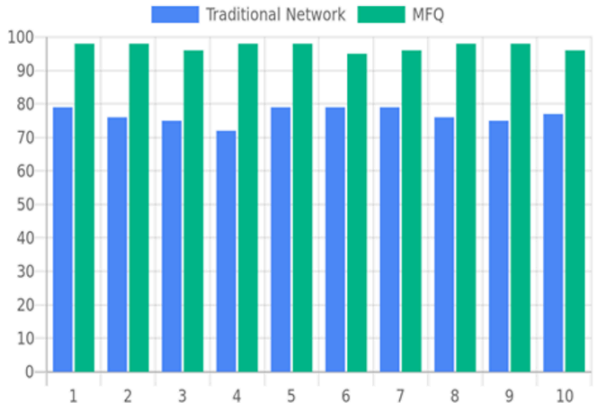


**Fig. 4:** Comparison of overall link utilization

Analysis 2: We built upon the findings from Study 1 to investigate algorithmic congestion control. We focused on a single shortest path, specifically the path from s1-5, and set the available bandwidth to 5 Mbps for the duration of 20 sec. During the simulation, Datacenter c4 sent UDP packets to h1 with a bandwidth of 2 Mbps, while Datacenters c1, c2, c3, and c4 transmitted UDP packets to h2, each with a bandwidth of 1 Mbps. We considered two scenarios in this experiment. The first scenario involved using the real bandwidth without any queues to control congestion, meaning that traffic was sent based solely on the available bandwidth, without mechanisms to manage or mitigate congestion. In the second scenario, queues were implemented to actively manage congestion. This approach aimed to control the flow of traffic and optimize the use of available bandwidth, allowing for more efficient data transmission despite potential congestion. By comparing these two scenarios, we sought to evaluate the impact of congestion control mechanisms on network performance, particularly in terms of bandwidth utilization and the overall efficiency of the network.

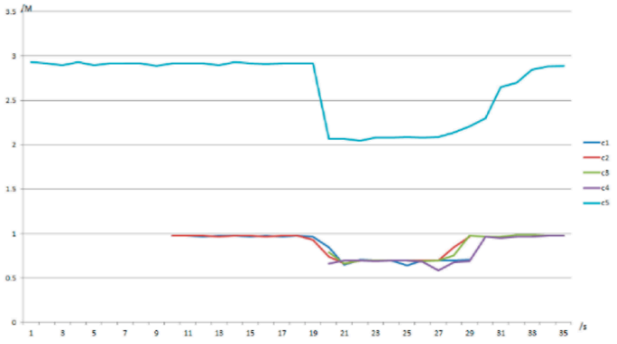

**Fig. 5:** Real-time bandwidth with no congestion control

In Figure 5, the route (s1, s4, s5) became saturated at 10 sec due to the combined flows from c1, c2 and c5. This saturation means that the route reached its maximum capacity and no more traffic could be accommodated without causing congestion. At 20 seconds, packet requests from c3 and c4 were introduced into the network. If congestion management queues are not utilized, the high-priority packets from c5 will be sent without any control mechanism in place. This can cause significant issues because the high-priority packets from c5 would directly compete with other ongoing flows in the network, leading to packet delays, packet drops, or increased retransmissions for lower-priority flows. Essentially, without the use of queues to manage

congestion, the prioritization of certain traffic (such as the high-priority packets from c5) would disrupt the performance of other flows, causing unfair bandwidth distribution and potentially reducing the overall throughput of the network. Congestion management queues are crucial in such situations because they can help ensure that traffic flows are managed appropriately, allowing for better prioritization and ensuring that congestion does not negatively impact the performance of the entire network.

In Figure 6, when congestion control queues are implemented, the network can effectively manage the flow of traffic to prevent congestion from negatively affecting performance. As the packet requests from c3 and c4 arrive, the system evaluates the bandwidth available on the path. With congestion control in place, the flow from c5, which had previously been contributing to network congestion, can be rescheduled.

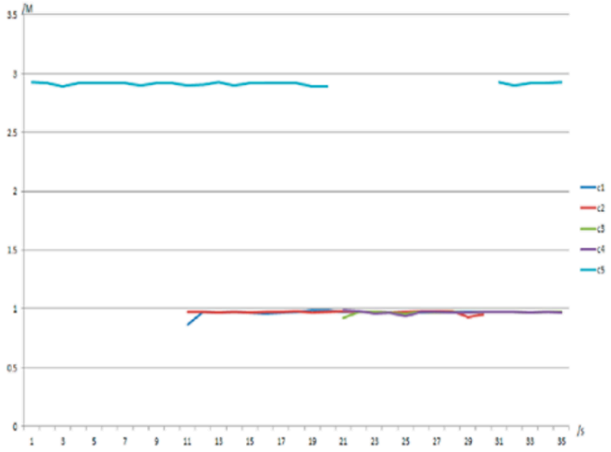


**Fig. 6:** Real-time bandwidth congestion control

The congestion management mechanism works by dynamically assessing the remaining available bandwidth on the path. When the congestion from other flows (such as those from c3 and c4) is detected, the system prioritizes the efficient use of available resources. It identifies that the flow from c5 can be shifted to a path that has higher transmission capacity, thus preventing the flow from c5 from contributing to further congestion. This reassignment ensures that the c5 flow continues without being impacted by the congestion from other flows.

By rescheduling the c5 flow to a path with greater capacity, the congestion control queues help maintain smooth data transmission and ensure that each flow is transmitted efficiently. This not only minimizes packet delays but also prevents packet drops or the need for retransmissions, ensuring the network operates at its optimal performance. This approach highlights how congestion control can dynamically adapt to network conditions, improving overall throughput and fairness in resource allocation.

## Discussion

The flow distribution of network traffic is not uniform and is not always even. Network traffic might more than quadruple during peak hours when compared to a normal load connection. To balance the connection load using the shortest path algorithm, conventional routing protocols, on the other hand, route and forward flows without pushing off flows. However, in order to balance the connection load using the shortest path algorithm, classical routing protocols forward and route flows without pushing off flows. To meet the bandwidth requirements during peak hours, we must purchase 2–3× bandwidth and upgrade to large-scale routing equipment. Network administration, upkeep, and service, As a result, fees are increased significantly, which usually leads to bandwidth waste during ordinary business hours. To measure and monitor network performance, the system incorporates two essential elements. Important connection data, such as bandwidth consumption and congestion levels across different network links, are recorded by the first module. When evaluating the performance and scalability of a network, this real-time data is essential. The second module records information about incoming and outgoing traffic at the network nodes by tracking packet-in and packet-out messages within the simulation protocol. These messages give granular visibility into the system's operation by offering comprehensive insights into packet movement throughout the network. Traffic control and real-time detection are made possible by the combination of these components. The system can quickly identify the need for changes and take appropriate action when new data is received or when a flow has to be scheduled. This procedure guarantees that available bandwidth is used effectively and congestion is kept to a minimum. Our MFQ-based system can improve network performance comparatively with the traditional network by prioritizing flows, optimizing resource allocation in real-time, and dynamically adapting to traffic fluctuations by utilizing the Floodlight controller's capabilities. Traditional Network: h1-c1: Time 0-1s: Bandwidth is 106 kB, which suggests a relatively low transfer rate at the beginning, possibly due to initial setup or low demand for this connection. Time 1-2s: Bandwidth drops significantly to 11.8 kB, indicating a large decrease in available bandwidth, which might indicate network congestion or the flow being deprioritized. Time 2-3s: It remains low at 11.8 kB, showing consistent underutilization of the bandwidth for h1-c1, which could be due to congestion or limited demand. Time 3-4s: The transfer rate jumps to 153 kB, a significant increase, which could reflect a burst of traffic or improved network conditions. Time 4-5s: It drops again to 35.3 kB, showing some fluctuation in bandwidth. This could indicate temporary congestion or routing issues. Time 5-6s: The bandwidth increases to 82.3 kB, reflecting a stable transfer rate. Time 6-7s: It decreases again to 35.3

kB, showing potential instability or competition with other traffic. Time 7-8s: Bandwidth further drops to 23.5 kB, indicating continued congestion or lack of demand. Time 8-9s: Drops again to 11.8 kB, maintaining a low transfer rate. Time 9-10s: Finally, bandwidth slightly increases to 23.5 kB, showing a bit of recovery. The conclusion of traditional network h1-c1 shows considerable fluctuations in transfer rates, which is typical when there is congestion, network instability, or competition for bandwidth. Bandwidth availability can vary widely depending on network conditions. When its comparison with the MFQ-based system stable bandwidth remains mostly around 1.94-1.95M for the majority of the time ($1^{st}$-$8^{th}$ sec), indicating a stable and reliable connection. The bandwidth drops slightly to 1.93M at the $6^{th}$ sec, but this is still within a narrow range, suggesting that there are small fluctuations but no significant issues. At the $9^{th}$ sec, there's a small peak to 1.97M, showing a brief increase in throughput. This could be due to network conditions improving temporarily, or increased demand for the h1-c1 flow during that second. At the $10^{th}$ sec, there is a drop to 1.87M, which is the lowest value in the sequence. This could reflect a temporary decrease in bandwidth availability, possibly due to congestion, a sudden change in the network load, or a competition for resources. When comparing MFQ data with the Traditional Network data for h1-c1, you can observe that the MFQ is much more stable and performs at consistently high levels (1.94-1.97M), with only a minor decrease at the end. In contrast, the Traditional Network had much greater fluctuations (ranging from 11.8-153 kB), suggesting that the Traditional Network had issues such as network congestion, routing problems, or competition for bandwidth, which impacted its consistency. However, the MFQ model addresses the network congestion and minimizes the traffic scheduling challenges.

## Conclusion

Our proposed MFQ scheme can be easily implemented to address the issue of low resource utilization in connection-heavy environments like live streaming and online video data centers, as well as in existing data center networks. The scheme effectively resolves the traffic scheduling challenges, operating within certain limitations that do not compromise transmission efficiency. First, because our scheme is based on Software Defined Networking (SDN), it is not compatible with traditional network routing protocols or hybrid network environments. Second, as rescheduling and retransmission occur within the routing protocol, packet loss is minimized, ensuring that packets are successfully delivered to their destinations. This results in a model that offers high throughput and low latency while efficiently utilizing available resources. Third, incorporating more flexible hierarchical control could further enhance the experimental results. In the future, we plan to address the energy efficiency issue by focusing on traffic coordination and congestion management within the network.

## Acknowledgment

Thank you to the publisher for their support in the publication of this research article. We are grateful for the resources and platform provided by the publisher, which have enabled us to share our findings with a wider audience. We appreciate the efforts of the editorial team in reviewing and editing our work and we are thankful for the opportunity to contribute to the field of research through this publication.

## Funding Information

This study was supported by Multimedia University, Malaysia.

## Author's Contributions

**R. Shanthy**: Conceived the research idea, designed the methodology, evaluated findings, and led the writing of the manuscript.

**K. Sutha**: Managed data collection and reporting, conducted the literature review, and reviewed the study for scientific accuracy.

**Joseph Emerson Raja**: Oversaw study execution and provided necessary resources, including personnel, funding, and equipment.

## Ethics

The authors declare no conflicts of interest of relevance to this topic.

## References

Alhilali, A. H., & Montazerolghaem, A. (2023). Artificial intelligence based load balancing in SDN: A comprehensive survey. *Internet of Things*, *22*, 100814. https://doi.org/10.1016/j.iot.2023.100814

Ali, S. T., Sivaraman, V., Radford, A., & Jha, S. (2015). A Survey of Securing Networks Using Software Defined Networking. *IEEE Transactions on Reliability*, *64*(3), 1086-1097. https://doi.org/10.1109/tr.2015.2421391

Benson, T., Anand, A., Akella, A., & Zhang, M. (2011). MicroTE: fine grained traffic engineering for data centers. *Proceedings of the Seventh Conference on Emerging Networking EXperiments and Technologies*, 1-12. https://doi.org/10.1145/2079296.2079304

Cheng, J., Cheng, J., Zhou, M., Liu, F., Gao, S., & Liu, C. (2015). Routing in Internet of Vehicles: A Review. *IEEE Transactions on Intelligent Transportation Systems*, *16*(5), 2339-2352. https://doi.org/10.1109/tits.2015.2423667

Cui, L., Yu, F. R., & Yan, Q. (2016). When big data meets software-defined networking: SDN for big data and big data for SDN. *IEEE Network*, *30*(1), 58-65. https://doi.org/10.1109/mnet.2016.7389832

Curtis, A. R., Kim, W., & Yalagandula, P. (2011). Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. *IEEE INFOCOM 2011 - IEEE Conference on Computer Communications*, 1629-1637. https://doi.org/10.1109/infcom.2011.5934956

Dean, J., & Ghemawat, S. (2008). MapReduce. *Communications of the ACM*, *51*(1), 107-113. https://doi.org/10.1145/1327452.1327492

Ghaffarinejad, A., & Syrotiuk, V. R. (2014). Load Balancing in a Campus Network Using Software Defined Networking. *2014 Third GENI Research and Educational Experiment Workshop*, 75-76. https://doi.org/10.1109/gree.2014.9

Ghemawat, S., Gobioff, H., & Leung, S.-T. (2003). The Google file system. *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, 29-43. https://doi.org/10.1145/945445.945450

Greenberg, A., Hamilton, J., Maltz, D. A., & Patel, P. (2008). The cost of a cloud. *ACM SIGCOMM Computer Communication Review*, *39*(1), 68-73. https://doi.org/10.1145/1496091.1496103

Hindman, B., Konwinski, A., & Zaharia, M. (2011). Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center. *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*, 1-14.

Huang, X., Zeng, M., & Xie, K. (2021). Intelligent traffic control for QoS optimization in hybrid SDNs. *Computer Networks*, *189*, 107877. https://doi.org/10.1016/j.comnet.2021.107877

IEEE. (2010). OSDI (9th Usenix Symposium on Operating Systems Design and Implementation) advertisement. *IEEE Security & Privacy Magazine*, *8*(4), c4-c4. https://doi.org/10.1109/msp.2010.134

Kanagavelu, R., Mingjie, L. N., Mi, K. M., Lee, F. B.-S., & Heryandi. (2012). OpenFlow based control for re-routing with differentiated flows in Data Center Networks. *2012 18th IEEE International Conference on Networks (ICON)*, 228-233. https://doi.org/10.1109/icon.2012.6506562

Khurshid, A., Zhou, W., Caesar, M., & Godfrey, P. B. (2012). Veriflow: verifying network-wide invariants in real time. *ACM SIGCOMM Computer Communication Review*, *42*(4), 467-472. https://doi.org/10.1145/2377677.2377766

Kim, H., & Feamster, N. (2013). Improving network management with software defined networking. *IEEE Communications Magazine*, *51*(2), 114-119. https://doi.org/10.1109/mcom.2013.6461195

Lan, Y.-L., Wang, K., & Hsu, Y.-H. (2016). Dynamic load-balanced path optimization in SDN-based data center networks. *2016 10th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, 1-6. https://doi.org/10.1109/csndsp.2016.7573945

Li, J., Chang, X., Ren, Y., Zhang, Z., & Wang, G. (2014). An Effective Path Load Balancing Mechanism Based on SDN. *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, 527-533. https://doi.org/10.1109/trustcom.2014.67

Lin, P., Bi, J., & Wang, Y. (2015). WEBridge: west-east bridge for distributed heterogeneous SDN NOSes peering. *Security and Communication Networks*, *8*(10), 1926-1942. https://doi.org/10.1002/sec.1030

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., & Turner, J. (2008). OpenFlow. *ACM SIGCOMM Computer Communication Review*, *38*(2), 69-74. https://doi.org/10.1145/1355734.1355746

Nunes, B. A. A., Mendonca, M., Nguyen, X.-N., Obraczka, K., & Turletti, T. (2014). A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. *IEEE Communications Surveys & Tutorials*, *16*(3), 1617-1634. https://doi.org/10.1109/surv.2014.012214.00180

Sezer, S., Scott-Hayward, S., Chouhan, P., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., & Rao, N. (2013). Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine*, *51*(7), 36-43. https://doi.org/10.1109/mcom.2013.6553676

Shukla, P. K., Maheshwary, P., Subramanian, E. K., Shilpa, V. J., & Varma, P. R. K. (2023). Traffic flow monitoring in software-defined network using modified recursive learning. *Physical Communication*, *57*, 101997. https://doi.org/10.1016/j.phycom.2022.101997

Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). The Hadoop Distributed File System. *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, 1-10. https://doi.org/10.1109/msst.2010.5496972

Tavakoli, A., Casado, M., Shenker, S., & Koponen, T. (2009). Applying NOX to the Datacenter. *Proceedings of the Eighth ACM Workshop on Hot Topics in Networks (HotNets-VIII)*, 1-6.

Tootoonchian, A., & Ganjali, Y. (2010). HyperFlow: A distributed control plane for OpenFlow. *Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking*, 3. https://doi.org/10.5555/1863133.1863136

Tso, F. P., & Pezaros, D. P. (2013). Baatdaat: Measurement-based flow scheduling for cloud data centers. *2013 IEEE Symposium on Computers and Communications (ISCC)*, 000765-000770. https://doi.org/10.1109/iscc.2013.6755041

Xia, W., Wen, Y., Foh, C. H., Niyato, D., & Xie, H. (2015). A Survey on Software-Defined Networking. *IEEE Communications Surveys & Tutorials*, *17*(1), 27-51. https://doi.org/10.1109/comst.2014.2330903

Yu, Y., Lin, Y., Zhang, J., Zhao, Y., Han, J., Zheng, H., Cui, Y., Xiao, M., Li, H., Peng, Y., Ji, Y., & Yang, H. (2014). Field Demonstration of Datacenter Resource Migration via Multi-Domain Software Defined Transport Networks with Multi-Controller Collaboration. *Optical Fiber Communication Conference*, 1-3. https://doi.org/10.1364/ofc.2014.w1e.2

Zhang, C., Hu, J., Qiu, J., & Chen, Q. (2017). Reliable Output Feedback Control for T-S Fuzzy Systems With Decentralized Event Triggering Communication and Actuator Failures. *IEEE Transactions on Cybernetics*, *47*(9), 2592-2602. https://doi.org/10.1109/tcyb.2017.2668766