International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Hybrid Congestion Control Mechanism in Software Defined Networks

Reecha Sood [1], Dr. Sandeep Singh Kang [2]

Submitted: 26/09/2023 **Revised**: 16/11/2023 **Accepted**: 27/11/2023

Abstract Software-Defined Networking (SDN) has emerged as a promising paradigm to manage network traffic efficiently and provide enhanced performance. Queue management plays a critical role in SDN by effectively controlling the flow of packets and ensuring Quality of Service (QoS). We have implemented WFQ, which provides fairness and QoS guarantees. Assign weights to different flows based on their importance or priority, ensuring equitable distribution of network resources. WFQ dynamically adjusts transmission rates based on flow weights, preventing congestion, and maintaining optimal performance.Monitor queue lengths and implement policies to trigger congestion control measures when thresholds are exceeded. These measures may include queue length, goodput, or notifying source nodes to reduce their transmission rates. This strategy ensures efficient resource allocation, congestion control, and adherence to QoS requirements, resulting in a more robust and responsive SDN environment.

Keywords: *QoS, SDN, TCP, TCP Fairness, Congestion Window.*

1. Introduction

TCP faces several challenges in heterogeneous networks due to the diversity of network technologies, protocols, and requirements. Researchers are working on developing new TCP variants and queue management policies to address the queue problem, congestion, and other challenges to ensure efficient and reliable data transfer in such networks.Queues are used to temporarily store packets and manage their transmission based on various policies and priorities.In SDN, the network control is centralized, allowing for dynamic traffic management and fine-grained control over network resources. When it comes to managing heterogeneous traffic, SDN controllers can employ different queue management techniques to ensure efficient utilization of network resources and meet the quality of service (QoS) requirements of various traffic types.The SDN controller employs various queue management mechanisms to control the transmission of packets from each queue. Different algorithms can be used, such as First-In-First-Out (FIFO), Weighted Fair Queuing (WFQ), or Hierarchical Token Bucket (HTB). These algorithms ensure that packets are processed and transmitted according to the defined policies.By utilizing queues and employing intelligent queue management techniques, SDN controllers can effectively handle heterogeneous traffic, prioritize critical applications, ensure fair resource allocation, and optimize network performance based on specific requirements and policies.Weighted Fair Queuing (WFQ) is a queue management algorithm that can be used in Software-Defined Networking (SDN) to improve the performance of heterogeneous traffic. It provides fairness and quality of service (QoS) guarantees by assigning weights to different traffic flows, allowing for better resource allocation and congestion control.

Secondly, SDN based Wireless networks often struggle to provide consistent Quality of Services (QoS)[1]due to shared resources, interference, and limitations in the wireless medium. QoS issues can result in packet drops, delays, or prioritization conflicts, leading to TCP failures.TCP uses a congestion control algorithm that relies on detecting packet loss as an indication of network congestion. However, in networks with heterogeneous bandwidth and delay, packet loss can occur due to other reasons such as transmission errors, leading to inefficient use of available bandwidth[2]. Congestion control schemes are mechanisms used by network protocols to prevent or mitigate congestion in the network, which can lead to packet loss, delays, and reduced throughput.There are several types of congestion control algorithms, which can be broadly categorized into three categories:

1.1 AIMD (Additive Increase Multiplicative Decrease) algorithms: AIMD algorithms are the most widely used type of congestion control algorithms. They increase the sending rate of a flow by a small amount (additive increase) until congestion is detected, at which point they reduce the sending rate by a larger amount (multiplicative decrease). TCP Reno, TCP New Reno[3], and TCP Vegas are examples of AIMD algorithms.

1.2 Window-based algorithms: Window-based algorithms, such as TCP BIC (Binary Increase Congestion Control), adjust the size of the congestion window based on the amount of congestion in the network. These algorithms use a binary search to quickly converge on the optimal congestion window size[4].

[1] *Research Scholar, Chandigarh University Gharuan, Punjab, India*
reecha.coecse@cgc.edu.in
[2] *Research Scholar, Chandigarh University, Gharuan, Punjab, India*
sandeepkang.cse@cumail.in

1.3 Delay-based algorithms: Delay-based algorithms, such as TCP CUBIC (Compound TCP for TCP Vegas and TCP New Reno), use the Round-Trip Time (RTT) of packets to adjust the sending rate of a flow. These algorithms use a cubic function to adjust the congestion window size based on the RTT and the sending rate.

The best congestion control algorithm depends on the specific requirements of the network and the applications running on it. For example, AIMD algorithms are effective for networks with high levels of congestion, as they reduce the sending rate more aggressively in response to congestion. Window-based algorithms are more effective in networks with high-speed links and low levels of congestion. Delay-based algorithms are most effective in networks with high-latency links.

Overall, the choice of the best congestion control algorithm is dependent on the specific network characteristics and the goals of the congestion control mechanism. There is no one-size-fits-all algorithm that works best in all scenarios. Researchers continue to develop new congestion control algorithms to address the evolving needs of the network.

2. Motivation of Research

TCP Delayed Acknowledgment and Joint/Split Congestion Control[5] are two separate mechanisms used by TCP to improve network performance and congestion control.

TCP Delayed Acknowledgment [6]is a mechanism that helps reduce the number of acknowledgments sent by the receiver to the sender, by delaying the acknowledgment of received packets until either a certain amount of time has passed, or a certain number of packets have been received. This mechanism is used to reduce the overhead of sending acknowledgments and to improve the efficiency of the TCP protocol.

On the other hand, Joint/Split Congestion Control [7]is a mechanism used to improve congestion control in TCP. Joint Congestion Control is a technique that allows multiple TCP connections to share the same congestion control state, which can help reduce the amount of congestion in the network. Split Congestion Control, on the other hand, allows multiple TCP flows to have independent congestion control states, which can help avoid congestion collapse in the network.

Both Delayed Acknowledgment [8], [9]and Joint/Split [6]Congestion Control can help improve the performance of TCP in different ways. Delayed Acknowledgment helps reduce the overhead of sending acknowledgments, while Joint/Split Congestion Control helps improve congestion control in the network. However, these mechanisms are used for different purposes but in this research, we used TCP delayed acknowledgment and apply on receiver based SDN networks. The objective of this research is to maintain congestion window before expiration time of acknowledgment.

Table 1: Comparison of Slow Start, Fast Recovery and Retransmit and Selective Acknowledgement

Schemes	Network Overhead	Transmission Rate	Recovery of Loss Packets	Retransmission of Packets	Buffer Management
Slow Start	No	Slow	Yes	Yes	Yes
Fast Recovery	Yes	Fast	Yes	No	No
Fast Retransmit	Yes	Fast	Yes	No	No
Selective Acknowledgment	No	-----	No	------	-----

3. Research Gap

SDN allows for dynamic network management and flexibility, but large-scale deployments can present scalability challenges. As the network grows and the number of flows and devices increases, QoS and queue management mechanisms may struggle to handle the high traffic demands efficiently, resulting in performance degradation or failures[10].Adaptive queue management techniques, such as Active Queue Management (AQM) algorithms, can be employed in SDN environments. AQM algorithms[11], such as CoDel (Controlled Delay) or PIE (Proportional Integral controller Enhanced), dynamically adjust queue lengths and drop or mark packets based on congestion signals. These algorithms help ensure fair sharing of network resources and prevent network degradation during high traffic situations.SDN enables proactive congestion detection [12]and avoidance mechanisms. By monitoring network performance metrics, such as link utilization or packet loss rates, SDN controllers can detect congestion hotspots[13]. Queue management schemes can then take action to avoid congestion, such as redirecting flows, dynamically adjusting queue sizes, or applying congestion control algorithms (e.g., RED - Random Early Detection)[14].SDN enables centralized control and programmability, allowing for traffic engineering and path optimization. Queue management policies (the aim of this paper) can leverage SDN controllers to dynamically adjust flow paths, optimize traffic distribution, and direct traffic away from congested links or resources. This helps in effectively handling high traffic scenarios and maximizing network performance.Queue management in SDN allows for dynamic adjustments of queue parameters based on network conditions. SDN controllers can adaptively adjust queue lengths, thresholds, or drop/mark probabilities in response to changing traffic patterns or congestion levels. Dynamic queue adjustments ensure efficient utilization of queue resources and minimize the impact of high traffic on network performance.Queue management in SDN involves prioritizing traffic based on QoS requirements or flow characteristics. Flows with higher priority, such as real-time or critical applications, are given preferential treatment in terms of bandwidth allocation, queuing delays, or packet drop policies. By prioritizing critical traffic, high-priority flows can maintain their performance even during periods of high traffic.

4. Problem Formulation

Queue management policies [15]can be applied in Software-Defined Networking (SDN) networks using the OpenFlow protocol, which is a key component of many SDN implementations. OpenFlow provides a standardized interface between the control plane and the data plane in a network, enabling network administrators to control network behaviour by programming network devices, such as switches and routers.

In an SDN network, queue management policies can be applied by configuring the queue management parameters of OpenFlow switches. This can include setting queue thresholds, buffer sizes, and drop policies. Queue management policies can be applied to different types of traffic flows, such as TCP or UDP, based on the flow's Quality of Service (QoS) requirements.

OpenFlow switches maintain one or more queues for each port. These queues can be configured with different queue management policies, such as Random Early Detection (RED) [16]or Weighted Random Early Detection (WRED), which are used to manage congestion by selectively dropping packets.

Queue management policies can also be applied dynamically by the SDN controller. For example, if the controller detects congestion on a particular link, it can send instructions to the OpenFlow switches to adjust the queue management policies for that link to prevent congestion.WFQ ensures fair allocation of network resources among different traffic flows. Each flow is assigned a weight that represents its relative importance or priority. The weights determine the amount of bandwidth allocated to each flow. This fairness mechanism prevents one flow from monopolizing network resources, ensuring that all traffic types receive a fair share of available bandwidth.By employing Weighted Fair Queuing (WFQ) in SDN, network administrators can achieve better performance, fairness, and QoS guarantees for heterogeneous traffic. Additionally, SDN networks can use traffic engineering techniques to route traffic around congested links. This can be done by configuring OpenFlow switches to redirect traffic flows to less congested paths in the network. Overall, queue management policies can be applied in SDN networks using OpenFlow switches to manage congestion and improve the performance of the network. The ability to dynamically apply queue management policies based on changing network conditions is one of the key benefits of SDN.

5. Related Work

Base station could act as a strategic point for hosting servers is in a remote or rural area where there is limited connectivity to the core network[7]. The aggregated flow routed between joint and split points of the network in a user agnostic manner. In general, base stations are not designed to be hosting servers, and deploying servers in a base station could introduce additional complexity and risk. However, in certain scenarios where connectivity to the core network is limited or disrupted, a base station with built-in servers could provide a valuable strategic point for hosting critical services.

Single long Transmission Control Protocol (TCP) flow can be transmitted as aggregated traffic [17]. TCP is a protocol that provides reliable, ordered, and error-checked delivery of data between applications running on hosts connected via a network. TCP divides the data into segments, which are transmitted as packets over the network.When multiple packets from a single long TCP flow are transmitted over the network, they may be aggregated together by intermediate network devices, such as switches or routers. This aggregation can help reduce the number of packets transmitted over the network and improve network efficiency.However, while aggregating TCP flows can improve network efficiency, it can also introduce some challenges. For example, if packets from multiple TCP flows are aggregated together, it may be more difficult to manage Quality of Service (QoS) parameters for individual flows, such as bandwidth allocation or priority.

Researchers compare the existing and previous congestion control algorithms using TCP flow completion time [18]. TCP completion time refers to the amount of time it takes for a TCP flow to complete, from the start of the communication to the end. TCP completion Time is suitable for both short and long TCP flows. Completion time of large TCP flows is high as compared to short TCP flows. To reduce the completion time for long TCP flows, various techniques can be used, such as congestion control algorithms, window scaling, and selective acknowledgments. These techniques help to ensure that the flow can transmit data at an optimal rate while minimizing the impact of network issues.

Researchers work on contention-based window policy and presented HBAB algorithm [19]. HBAB defines three variables that identify the channel state and update the contention window. History-based adaptive algorithms rely on historical data to predict future network conditions and adjust transmission rates accordingly. While these algorithms can be effective in some scenarios, they may not be able to react quickly enough to changes in network congestion, which can result in suboptimal performance.Moreover, history-based algorithms require storing and processing large amounts of historical data, which can be resource-intensive and may not be practical for use in congestion-based algorithms.

Nonlinear ordinary differential equations (ODEs) can indeed be helpful in queue management systems. Queue management involves analysing and optimizing the flow of entities (such as customers, requests, or tasks) through a system, minimizing wait times, improving efficiency, and maximizing resource utilization. Nonlinear ODEs can capture complex dynamics and interactions within the queueing system, allowing for a more accurate representation of its behaviour [14].Overall, nonlinear ODEs provide a powerful mathematical framework for analysing and managing queues.

Active queue management helps to decide on the transmission rate in TCP congestion control approaches.The AIMD algorithm is used to solve issues such as retransmission timeout scheme which is also beneficial to the researcher. The researchers presented algorithms that implement queue state and load state in TCP. The drop probability can also be calculated with the help of the proposed scheme.The queue packet arrival rate is calculated, and the overall packet arrival rate is estimated. Reference variable is used to control thequeue length. The receiving state updates the length of the current queue. Improper estimation causes unstable network and affects the overall performance of the network [20].

6. Proposed Algorithm

AQM can be viewed as the queue length for single bottleneck (B) and double bottleneck. A single bottleneck capacity of µpackets per second and can be doubled for double bottleneck. The main drawback of AQM is that it cannot control the queue which is causing it to be full [21].

To normalize the size of the buffer, we take

$$B= \frac{\boldsymbol{B}}{\mu \boldsymbol{T+1}} \text{-----} \qquad (1)$$

Thus, we can estimate the delay calculation of the barrier. Packet marking behaviour is also a key point of this proposed approach as we can better test the queue length. The propagation delay constant for a single bottleneck containing B packets is It is easy to detect packet loss with the help of AIMD (Additive Increase and Multiple loss) algorithm. When the sender is not acknowledged, it means that the delay increases, and consequently the performance decreases. The main objective of this paper is to improve the performance of SDN networks and consequently calculate the delay with the round-trip time parameter.

$$\text{Wpipe}= \mu\mathbf{T} + \mathbf{B} \text{-----} \qquad (2)$$

Now, the variable B stores the packet in the queue before processing of the packets. The number of unacknowledged packets were identified by the sequence number. In this mathematical scenario, we take a steady state where the mobile nodes were not moving, and window W increases by one if all packets were constantly buffered in the queue at every RTT. We take RTT as 3 milliseconds (ms) in a steady state and applied 3-dupack policy during transmission of packets in the network. Furthermore, we are estimating the new queue length based on window probability. The probability of window taken as [0,1] and length of the queue is represented as q. we have also take the threshold value that helps to identify the probability of packet loss, queue length and estimation of empty packets. The threshold value areMin_T, Max_TandMax_P, these three variables indicates the queue length.

The probability of $\text{Max}_P$is the drop probability parameter and if the value of q is higher than Max_Pthen more chances to be congestion in the networks and also reverse the performance of the networks. During the setup of the network, the variable $w_q$taken as either positive or negative. The new formulated equation is represented as:q_{new}

$$\mathbf{qnew} = (\mathbf{1} - \mathbf{wq}).\mathbf{q} + \mathbf{wq}.\mathbf{q} \text{-----} \qquad \mathbf{(3)}$$

from the equation (3), if the negative value come the equation that means chances of buffer overflow in SDN network otherwise data transferring in steady way. A store and forward approach can be implemented in software defined networks (SDNs) that have not been used in prior techniques as evidenced in research papers[15], [16], [20], [22]–[24]. By sending L bits transmitted over a link with transmission rate R, a packet is represented over N links with P, and the L/R transmission of packets is represented as:

$$\boldsymbol{Tx} = \boldsymbol{P}.\boldsymbol{N}.\frac{\boldsymbol{L}}{\boldsymbol{R}}\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (4)$$

From equation 5, X has activated the retransmission policy before initiating packet loss in software-defined networks. The outgoing approach manages the queue length by calculating the length of the old queue and identifying the length of the new queue which is derived from Equation 3.

$$\boldsymbol{X} = \frac{\boldsymbol{avgq-minT}}{\boldsymbol{maxT-minT}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (5)$$

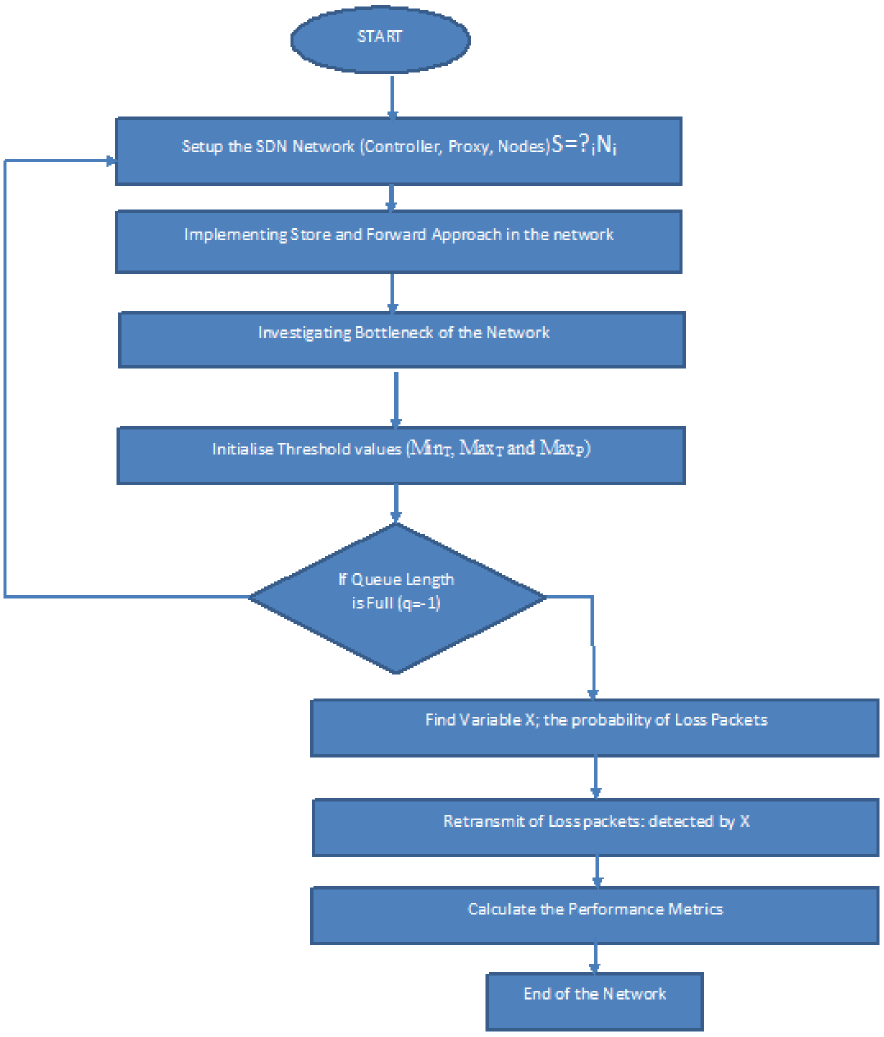
START
Setup the SDN Network (Controller, Proxy, Nodes)S=?¡Nᵢ
Implementing Store and Forward Approach in the network
Investigating Bottleneck of the Network
Initialise Threshold values (Min_T, Max_T and Max_P)
If Queue Length is Full (q=-1)
Find Variable X; the probability of Loss Packets
Retransmit of Loss packets: detected by X
Calculate the Performance Metrics
End of the Network

Fig 1 Proposed Framework

In an SDN architecture, the SDN controller is responsible for managing the flow of traffic through the network and can use TCP connections to facilitate communication between devices. For example, the controller may use a TCP connection to establish a communication channel between a Controller and a Proxy, or to route traffic between two devices on the network. TCP connections in an SDN architecture can be used to transmit a variety of types of data, including audio, video, and text. The SDN controller can use information about the type of data being transmitted and the devices involved to make decisions about how to route the traffic and allocate resources within the network.

A proxy is a device or software program that acts as an intermediary between two other devices or programs, forwarding requests and responses between them. An SDN proxy is a device or software program that combines the functions of both an SDN controller and a proxy. It is responsible for managing the flow of traffic through the network and forwarding traffic between different parts of the network. In an SDN architecture, the SDN proxy is often used to connect different parts of the network that are controlled by different SDN controllers, or to provide an interface between the SDN network and non-SDN devices.

In a SDN (software-defined networking) architecture, a queue is a data structure that stores packets of data that are waiting to be transmitted over the network. A queue can be used to store

packets that are waiting for a specific resource to become available (such as a link or a router) or packets that are waiting to be processed by the network control plane.

In SDN architecture, the SDN controller is responsible for managing the flow of traffic through the network and can use queues to regulate the flow of traffic. For example, the controller may use a queue to buffer incoming packets when the network is congested, or to prioritize certain types of traffic over others. In addition to managing traffic flow, queues can also be used for other purposes in an SDN architecture, such as monitoring network performance or providing information to network administrators.

To map the network, the SDN controller gathers information about the network topology, including the locations and capabilities of different devices on the network and the links between them. This information is used to create a map of the network, which can be used to determine the most efficient path for traffic to take to reach its destination.

To find a path through the network, the SDN controller uses algorithms and data structures (such as routing tables and flow tables) to determine the best route for traffic to take based on factors such as network congestion, available resources, and the requirements of the traffic. The controller can then use this information to update the network configuration and direct traffic to follow the chosen path.

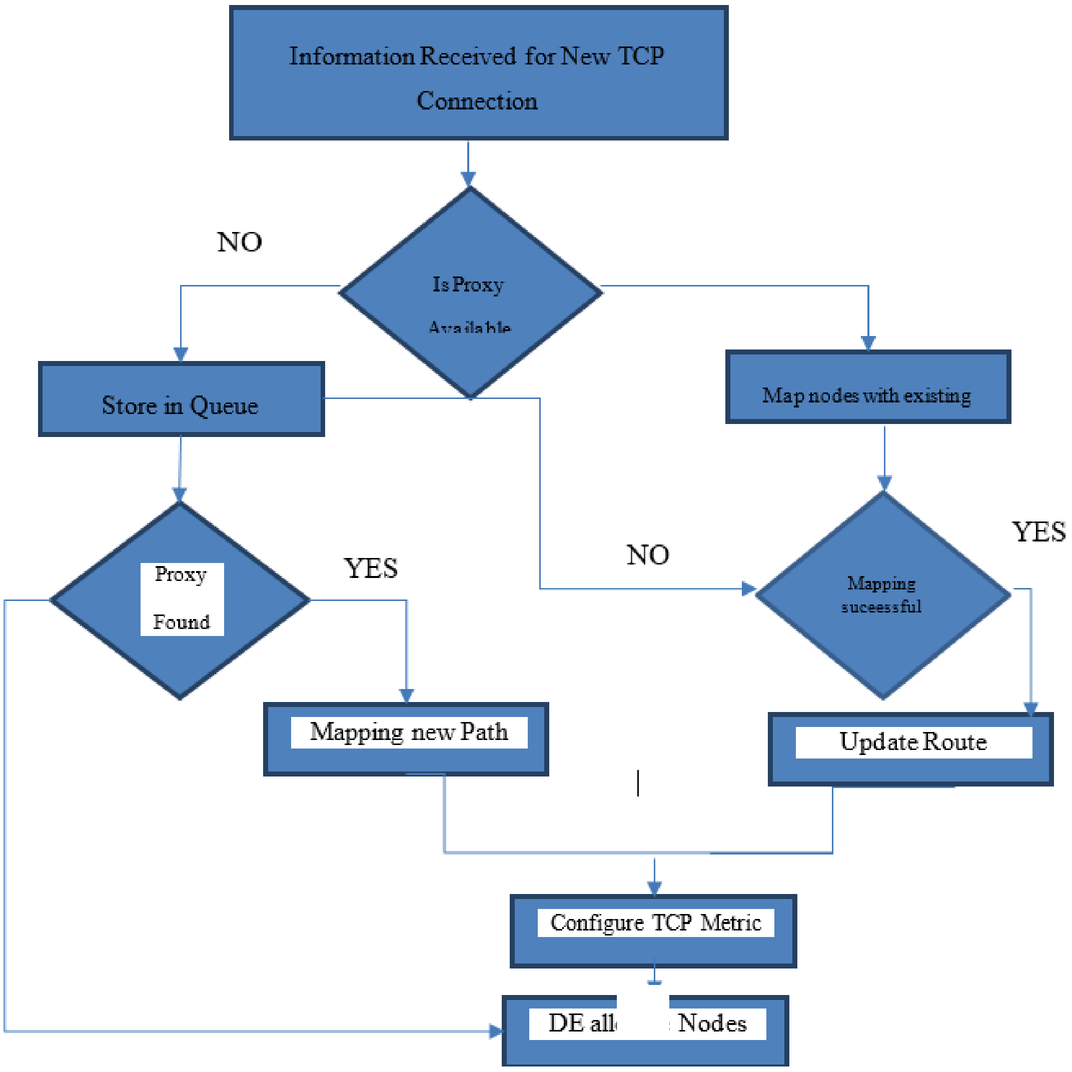

Fig 2: Proxy Configuration in Proposed Scenario

To deallocate nodes in an SDN architecture means to release them from their assigned tasks or traffic forwarding responsibilities and make them available for other uses. This can be done for a variety of reasons, such as to balance the load on the network, to remove a node that is no longer needed, or to reconfigure the network for a new set of requirements.

To deallocate nodes in an SDN architecture, the SDN controller must update the network configuration to remove the node from its assigned tasks or traffic forwarding responsibilities. This may involve updating flow tables, routing tables, or other data structures used to manage the flow of traffic through the network.

7. Performance Evaluation and Simulation Setup

This section shows the network topology of Software Defined Networks. The performance of SDN is implemented in NS2 simulator with TCP, internet services. The following values will show in the Table 2.

TABLE 2: Simulation Parameters of SDN-Queue Management Policy [7]

Parameter	Value
NS Version	2.34
Packet Size	1500
Queue Length	8
Buffer Size	35
Traffic Pattern	CBR, FTP
Transmission Range	1.5m
IEEE	802.11

TCP traffic aggregates the behaviour of SDN network and generate traffic with constant interval $N.\frac{L}{R}$ that specifies delay (d). All nodes are connected to each otherthrough controller and proxy. In network topology, the source node connected to destination node through bottleneck link that stream the packets from proxy node to another proxy node. Proxy node executes the queue management algorithm before it is configure in NS2. WFQ incorporates congestion control mechanisms to prevent network congestion and ensure smooth traffic flow.

When congestion occurs, Weighted Fair Queuing (WFQ) dynamically adjusts the rate at which packets are transmitted from each flow based on their assigned weights. Flows with higher weights are allowed to transmit more packets, while flows with lower weights are restricted. This adaptive behaviour helps alleviate congestion and maintain optimal network performance.

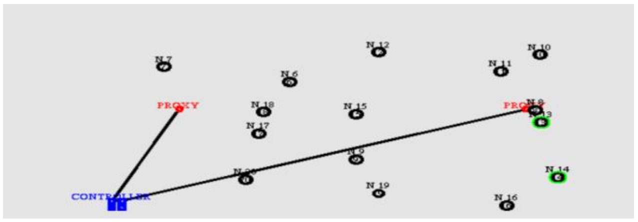

Fig 3: SDN based Simulation Scenario represented in NS2

8. Results and Discussions

A.Queue Length

Queue length refers to the number of packets or data units waiting in a queue to be processed or transmitted. In SDN, queues are used to manage traffic and temporarily store packets during congestion or when resources are temporarily unavailable. Queue length provides an indication of the level of congestion in the network. A longer queue length suggests a higher amount of buffered traffic, which may lead to increased delays and potential performance issues. Monitoring and managing queue length is crucial to avoid congestion and maintain efficient network performance.The threshold value (as suggested in flowchart) for queue length is important to prevent congestion and maintain desired QoS levels by triggering congestion avoidance mechanisms w/hen the queue exceeds the specified limit. The proposed hybrid algorithm having lesser average queue length than TCP protocol. Therefore, the proposed algorithm has performed better than existing protocol.

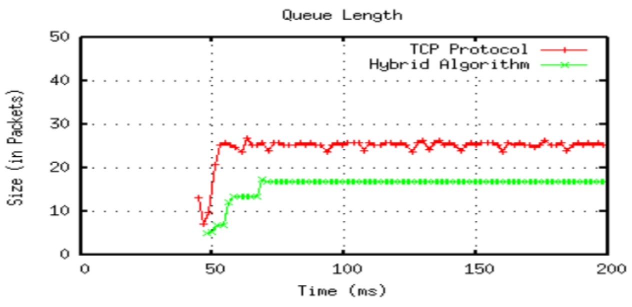

Fig 4: Comparison of TCP and Hybrid Algorithm

B.Goodput

With SDN-based Networks, TCP traffic increases linearly using the proposed method. The total traffic shown by Goodput in terms of traffic and performance is used to determine the basic performance of TCP and its techniques. The 1.5 Mbps connection specified in the suggested Switch 1 and Switch 2 models connects to the server. This proposed model demonstrates its performance using the NS2 simulator with 96% confidence in the current study. During inspection it was found that throughput increases linearly with the number of flows. On the other hand, due to the wide variety of traffic observed during the simulation, the performance of the TCP splitting method degrades by 1% compared to the proposed study. Temporary traffic has also been found to degrade the overall performance of regular TCP.

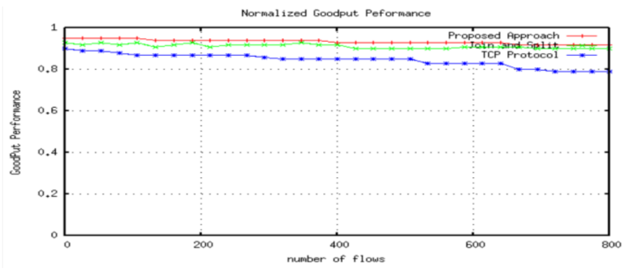

Fig 5: Goodput Performance

C.Congestion Window

The following diagram is based on the NS2 simulation shown in Figure 5.2. It shares common outages with the 5 Mbps link between proxy and nodes. The RTT parameter is fixed in the random-based model, and the value of RTT is set to 110. The completion time for each receiver is different and is shown in the simulation time. The comparison is shown for TCP, join split, and the proposed solution, starting at 0.1 and ending at 1.2 seconds. The growth of the congestion window decreases as the simulation time increases. In each subflow, the size of the congestion window starts from zero and increases by one. Performing the TCP three-way handshake process for each algorithm, it is observed that the proposed model better supports the model based on the SDN multi-way stochastic model and degrades the TCP performance of the entire network due to the unreserved window size. Furthermore, in the average random topology of the algorithm, the window size is doubled, and the packets are shrunk due to the overcrowded window. When the simulation time reaches 50 times, the proposed solution keeps dropping packets and congestion windows is reduced due to the large number of sub flows. Another reason is that the previous window remains unchanged, so some dips are observed in the chart.

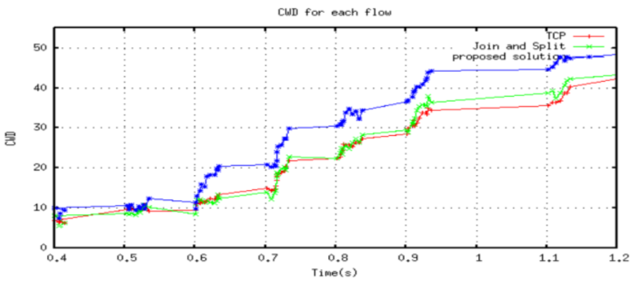

Fig 6: Congestion Window (cwd) for each flow

9. Conclusion

This paper ensures that high-priority flows in network scenario and get the necessary resources while still allowing lower-priority flows to utilize the available bandwidth. This efficient resource allocation improves overall network performance and maximizes the utilization of network resources.The Proposed Algorithm provides traffic isolation by treating each flow as a separate entity. Flows are processed and transmitted independently, which prevents one flow from adversely impacting others. This isolation helps to contain the effects of bursty or high-bandwidth flows, ensuring that they do not disrupt the performance of other flows in the network. Additionally, proposed algorithm is implemented in the proposed scenario that can handle a significant number of flows efficiently due to its weighted allocation scheme. It allows fine-grained control over each flow, enabling SDN controllers to manage and prioritize heterogeneous traffic effectively.

References:

[1] V. A. Tafti and A. Gandomi, "Performance of QoS Parameters in MANET Application Traffics in Large Scale Scenarios," *World AcadSciEngTechnol*, no. 72, pp. 857–860, 2010.

[2] Y. Li, S. Lei, X. You, H. Zhuang, and K. Sohraby, "Performance of TCP in intermittently connected wireless networks: Analysis and improvement," in *GLOBECOM - IEEE Global Telecommunications Conference*, 2010, pp. 1–6. doi: 10.1109/GLOCOM.2010.5684314.

[3] S. Fowler, M. Eberhard, and K. Blow, "Implementing an adaptive TCP fairness while exploiting 802.11e over wireless mesh networks," *International Journal of Pervasive Computing and Communications*, vol. 5, no. 3, pp. 272–294, 2009, doi: 10.1108/17427370910991857.

[4] S. Jasuja and P. Singh, "Appraisement of IEEE 802.11s based Mesh Networks with Mean Backoff Algorithm," *International Journal of Modern Education and Computer Science*, vol. 7, no. 10, pp. 20–26, 2015, doi: 10.5815/ijmecs.2015.10.03.

[5] W. Guo, V. Mahendran, and S. Radhakrishnan, "Join and spilt TCP for SDN networks: Architecture, implementation, and evaluation," *Computer Networks*, vol. 137, pp. 160–172, 2018, doi: 10.1016/j.comnet.2018.03.022.

[6] J. Chen, M. Gerla, Y. Z. Lee, and M. Y. Sanadidi, "TCP with delayed ack for wireless networks," vol. 6, pp. 1098–1116, 2008, doi: 10.1016/j.adhoc.2007.10.004.

[7] W. Guo, V. Mahendran, and S. Radhakrishnan, "Join and spilt TCP for SDN networks: Architecture, implementation, and evaluation," *Computer Networks*, vol. 137, pp. 160–172, 2018, doi: 10.1016/j.comnet.2018.03.022.

[8] S. Tcp, W. Tcp, and D. D. Protocol, "TCP-aware link layer based methods," 2007.

[9] N. H. Vaidya, M. N. Mehta, C. E. Perkins, and G. Montenegro, "Delayed duplicate acknowledgements: A TCP-Unaware approach to improve performance of TCP over wireless," *WirelCommun Mob Comput*, vol. 2, no. 1, pp. 59–70, 2002, doi: 10.1002/wcm.33.

[10] H. M. Syed, K. Das, and M. Devetsikiotis, "TCP performance and buffer provisioning for internet in wireless networks," in *IEEE International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems - Proceedings*, 1999, pp. 48–55. doi: 10.1109/mascot.1999.805039.

[11] No, V. S. J. Prakash, D. I. G. Amalarethinam, E. G. Dharma, and P. Raj, "CASE STUDY & SURVEY REPORT Available Online at www.ijarcs.info QoS Congestion Control AQM Algorithms : A Survey," vol. 2, no. 4, pp. 38–41, 2011.

[12] R. Fischer e Silva and P. M. Carpenter, "TCP

proactive congestion control for east-west traffic: The marking threshold," *Computer Networks*, vol. 151, pp. 1–11, 2019, doi: 10.1016/j.comnet. 2019.01.002.

[13] Y. Lu and S. Zhu, "SDN-based TCP congestion control in data center networks," *2015 IEEE 34th International Performance Computing and Communications Conference, IPCCC 2015*, 2016, doi: 10.1109/PCCC.2015.7410275.

[14] Agrawal and F. Granelli, "Redesigning an Active Queue Management System," 2004.

[15] M. N. Uddin, M. Rashid, M. Mostafa, S. Salam, N. Nithe, and S. Z. Ahmed, "Automated Queue Management System," *Type: Double Blind Peer Reviewed International Research Journal Publisher: Global Journals Inc*, vol. 16, 2016, [Online]. Available: http://creativecommons.

[16] M. Alkharasani, M. Othman, A. Abdullah, and K. Y. Lun, "An Improved Quality-of-Service Performance Using RED's Active Queue Management Flow Control in Classifying Networks," *IEEE Access*, vol. 5, pp. 24467–24478, 2017, doi: 10.1109/ACCESS. 2017.2767071.

[17] W. Guo, V. Mahendran, and S. Radhakrishnan, "End-User Agnostic Join and Fork Framework for TCP Flows in SDN," *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pp. 616–617, 2017, doi: 10.1109/CCNC. 2017.7983192.

[18] G. Luan, "Estimating TCP flow completion time distributions," *Journal of Communications and Networks*, vol. 21, no. 1, pp. 61–68, 2019, doi: 10.1109/JCN.2019.000006.

[19] M. Al-Hubaishi, T. Abdullah, R. Alsaqour, and A. Berqia, "E-BEB algorithm to improve quality of service on wireless Ad-Hoc networks," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 4, no. 7, pp. 807–812, 2012.

[20] J. Hong, C. Joo, and S. Bahk, "Active queue management algorithm considering queue and load states," *ComputCommun*, vol. 30, no. 4, pp. 886–892, Feb. 2007, doi: 10.1016/j.comcom. 2006.10.012.

[21] C. Long, B. Zhao, X. Guan, and J. Yang, "The Yellow active queue management algorithm," *Computer Networks*, vol. 47, no. 4, pp. 525–550, Mar. 2005, doi: 10.1016/j.comnet.2004.09.006.

[22] R. Brown, "Calendar Queues: A Fast {O(1)} Priority Queue Implementation for the Simulation Event Set Problem," vol. 31, no. 10, pp. 1220–1227, Oct. 1988.

[23] K. L. Tan and L.-J. Thng, "SNOOPy Calendar Queue," in *Proceedings of the 32nd conference on Winter simulation Orlando, Florida*, 2000, pp. 487–495.

[24] Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *Internetworking: Research and Experience*, vol. 1, no. 1, pp. 3–26, Jan. 1990.

[25] Mou, X., Sun, J., Zhong, Y., &Wo, T. (2023, July). HyCU: Hybrid Consistent Update for Software Defined Network. In *2023 IEEE International Conference on Joint Cloud Computing (JCC)* (pp. 86-92). IEEE.

[26] Lim, C. S., Tan, S. C., &Baderulhisham, N. Q. (2022, November). Energy And Congestion Awareness Traffic Scheduling In Hybrid Software-Defined Network. In *2022 IEEE International Conference on Computing (ICOCO)* (pp. 215-219). IEEE.

[27] AlShammari, W. M., &Alenazi, M. J. (2021). BL-Hybrid: A graph-theoretic approach to improving software-defined networking-based data center network performance. *Transactions on Emerging Telecommunications Technologies, 32*(1), e4163.

[28] Kadim, U. N., & Mohammed, I. J. (2020). A hybrid software defined networking-based load balancing and scheduling mechanism for cloud data centers. *Journal of Southwest Jiaotong University, 55*(3).

[29] R. Ying, W. K. Jia, Y. Zheng, and Y. Wu, "Fast Invalid TCP Flow Removal Scheme for Improving SDN Scalability," *2019 16th IEEE Annu. Consum. Commun. Netw. Conf. CCNC 2019*, 2019, doi: 10.1109/CCNC.2019.8651760.