NIPES - Journal of Science and Technology Research
www.nipes.org

Development of Dynamic Meter-Based Rate Limiter in SDN for Optimizing Multimedia Traffic Using P4 Language

Ahmad Enesi Siyaka[1], Salisu Aliyu[2], Sahabi Yusuf Ali[3]
Computer Science Department, Ahmadu Bello University (ABU), Zaria-Nigeria.
Email: abukhatheer@gmail.com[1], aliyusalisu@abu.edu.ng[2], sahabiali.yusuf@gmail.com[3]

Article Info

Keywords: *Software Defined Network (SDN), OpenFlow, Quality of Service, Multimedia traffic, Programing Protocol-Independent Packet Processors (P4) Language.*

Received 18 January 2025
Revised 26 March 2025
Accepted 15 April 2025
Available online 19 May 2025

https://doi.org/10.37933/nipes/7.2.2025.15

eISSN-2682-5821, pISSN-2734-2352

Abstract

Software Defined Networking (SDN) is a new paradigm in the networking environment that decouples the control plane from the data plane and introduces network agility and flexibility. This new technology has paved the way for innovative solutions to tackle the complexities of providing efficient Quality of Service (QoS) for multimedia traffic, which has grown exponentially at the same pace as the internet and internet-connected devices. Existing methods rely on legacy devices where QoS performance is poor due to their static metering and packet sampling approach. Previous researchers made significant contribution by introducing the Meter Band Rate Evaluation (MBE) mechanism which incorporates Band Rate Description Language (BRDL) and Band Rate Evaluator (BRE) components directly in the data plane over OpenFlow-based SDN. However, while MBE method effectively improved QoS for mission-critical traffic, it remained limited by the OpenFlow's configuration capabilities. This work leverages the new programming protocol-independent packet processors (P4) language to develop an indirect meter linked to the SDN Match-Action table with an efficient algorithm for dynamic, real-time adjustments of threshold rates and packet sampling interval. The proposed algorithm also integrates a packet recirculation strategy to manage oscillations in high-priority traffic, while low-priority traffic is distributed across N-distinct routes to maintain overall network efficiency. A genuine datacenter fabric topology is used to test the performance. The outcome of the experiments demonstrates the success of the approach in three out of the four metrics compared with MBE by maximizing throughput up to 30.14%, minimizing jitter by 46.93%, and reducing packet loss rate by 61.42%.

1.0 Introduction

Software-Defined Networking (SDN) is a new area in networking that is focused on making networks programable by decoupling the control plane from the data plane of the network [1]. The need for network programmability and dynamism in network management can be buttressed by the huge demand for Internet and Internet services as evident in the area of Internet of Things, Big Data, Cloud Computing services and Data Center Virtualization among others which had been on the increase at the same pace as the growth of network users, smart devices and sensors [2]. Therefore, one of the core issues in the area of networking in recent times is how to achieve network manageability more efficiently and effectively while meeting these huge demands in a manner that brings about tangible bottom-line results [2]. As a result of the Control Plane and Data plane decoupling in the SDN Paradigm, network operators can have total control of the network and associated network devices from a centralized SDN controller which maintains a global view of the topology through switch discovery, data link discovery, and host location discovery mechanisms [3].

This new approach to network architecture is already gaining momentum since it allows for the development of more dynamic, agile, and programmable networks [4]. The SDN paradigm postulates that the intelligence of networks lies in the presence of a logically centralized controller which makes the other network segments at the data plane to only be responsible for forwarding network traffics [5].

A notable advancement in SDN is the emergence of the Programming Protocol-independent Packet Processors (P4) language. This language has garnered significant interest due to its unique capabilities, allowing the programming of the data plane and exhibiting platform and protocol independence. The remarkable flexibility offered by P4 has the potential to elevate Quality of Service (QoS) control within SDN [6].

The development of scalable and efficient Quality of Service (QoS) frameworks in Software-Defined Networking (SDN) has been a subject of extensive research. [7] proposed a scalable QoS rate allocation framework for OpenFlow in SDN, addressing convergence issues found in the NUM-based algorithm. This framework incorporates an admission control scheme that verifies whether the upper values meet the capacity constraint, admitting flows accordingly and enhancing network scalability through decentralized algorithms running on multiple parallel controllers. However, this framework is formulated for flow-level dynamics and cannot be directly applied to packet-level dynamics, which involve bursty packet arrivals. Additionally, the reduced-NUM algorithm results in a sub-optimal solution.

[8] introduced an OpenFlow meter-based algorithm (OFMAQ) for IP multicasting in SDN, which dynamically provides bandwidth through a learning mechanism. This approach ensures that low-priority packets are redistributed among multiple routes, maintaining QoS for high-priority traffic. Despite its advantages, the learning approach results in a small amount of packet loss (0.05%-0.13%) due to dramatic changes in multimedia traffic.

[9] advanced the field with CATS, an SDN-based traffic scheduling algorithm designed for real-time congestion detection and control in video streaming scenarios. By aggregating flows into a larger flow and optimizing link utilization with a chaos genetic algorithm, CATS aims to improve overall QoS. However, its reliance on the OpenFlow Controller-Switch interaction for accurate and timely monitoring can lead to packet loss when this interaction is not feasible. [6] proposed a method for peak rate limitation and minimal bandwidth guarantee using a customized P4 pipeline for traffic engineering. This method allows best-effort traffic to consume unallocated bandwidth but is limited by its static bandwidth provisioning approach, reducing flexibility in handling diverse QoS requirements in complex networks. [10] contributed a policy-based QoS management framework for SDNs that utilizes Neural Networks to identify violating flows causing network congestion. Upon detection, the framework implements rerouting or rate limiting techniques. Despite its innovative approach, the complexity introduced by Neural Networks challenges real-time implementation in large-scale networks, and the model lacks proactive decision-making to minimize Service Level Objective (SLO) violations, which can negatively impact multimedia traffic.

Addressing congestion awareness, [11] introduced a two-segment scheduler model that improves network-wide congestion awareness through a per-node analytical approach using network calculus. This model accurately predicts key performance metrics such as throughput and delay but increases energy consumption due to its complex mathematical model, especially as network size increases. [12] further expanded traffic prioritization with the Multi-Color Marker (MCM), designed to ensure bandwidth guarantees and prioritize traffic in SDN virtual networks. MCM operates in two modes—Remaining Bandwidth Nonsharing (RBNS) and Remaining Bandwidth Sharing (RBS)—providing flexibility for network operators. However, the study does not fully address intra-VN packet prioritization, affecting fine-grained traffic control within VNs, and high-priority traffic struggles when the total exceeds the VN's guaranteed bandwidth.

[13] presented a P4-based bandwidth throttling mechanism implemented on a programmable switch, extending the BMv2 switch's capabilities through customized runtime software. Despite its contributions, this approach is less adaptive to other switch specifications and uses the $P4_{14}$ version, limiting its applicability with newer features of $P4_{16}$.

[14] introduced a novel solution to enhance QoS performance for streaming mission-critical video data in OpenFlow SDN networks. The Meter Band Rate Evaluation (MBE) mechanism enhances OpenFlow's native QoS capabilities but relies on predefined rates at BRDL, making maintenance difficult as network size and complexity increase. This approach requires modifications to existing OpenFlow components and integration of new elements, reducing adaptability to various other protocols without further changes.

Table 1: Summary of related works

S/N	Name of Author	Title of Paper	Year	Methodology	Strength	Weakness
1	Minh-Thuyen et al.	A Rate Allocation Framework for Multi-Class Services in Software-Defined Networks	2016	Scalable QoS rate allocation framework for OpenFlow in SDN	Addresses convergence issues found in the NUM-based algorithm. Additionally, enhances network scalability through decentralized algorithms	The framework is formulated and evaluated for flow-level dynamics and cannot be directly applied to packet-level dynamics, which involve

					capable of running on multiple parallel controllers.	bursty packet arrivals. They also admit that the reduced-NUM algorithm results in a sub-optimal solution.
2	Lin et al.	OpenE2EQoS: Meter-based Method for End-to-End QoS of Multimedia Services over SDN	2016	OpenFlow meter-based algorithm (OFMAQ) for IP multicasting in SDN.	Dynamically provides bandwidth through a learning mechanism. By multiplicatively increasing upcoming request by a factor of two; otherwise, additively decreased by 1% to gradually release the unused link capacity. Additionally, low-priority packets are redistributed among N routes.	Since the algorithm is based on a learning approach to model the multimedia traffic dynamics, it unavoidably results in a small amount of packet loss (0.05% 0.13%) due to the dramatic change of multimedia traffic.
3	Li et al.	Software Defined Traffic Engineering for Improving Quality of Service	2017	CATS, an SDN-based traffic scheduling algorithm for real-time congestion detection and control in video streaming.	Introduces the concept of aggregated elephant flow when flows share the congested link with the same source or destination DPID; or each flow is bigger than 1% of the congested link. Uses the chaos genetic algorithm to calculate the optimal distribution matrix for the aggregated elephant flow.	The reliance on OpenFlow Controller-Switch interaction for accurate and timely monitoring of link utilization might not always be feasible, potentially leading to some packet loss.
4	Yan-Wei et al.	P4-Enabled Bandwidth Management	2017	Peak rate limitation and minimal bandwidth guarantee using a customized P4 pipeline.	Allows best-effort traffic to consume the unallocated bandwidth.	Static bandwidth provisioning approach is employed, limiting the flexibility to handle diverse QoS requirements in complex networks.
5	Al-Jawad et al.	Policy-based QoS Management Framework for Software-Defined Networks	2018	Policy-based QoS management framework for SDNs using Neural Networks	Identifies violating flows causing network congestion. Upon detection of a policy violation, rerouting or rate limiting techniques are implemented.	No proactive intelligent decision-making for minimizing the number of Service Level Objective (SLO) violations. Availability of network resources determines admission or decline of service. The route manager chooses an alternate route or limit rate for violating traffic both of which is bad for multimedia traffic.
6	Darmani and Sangelaji	QoS-enabled TCP for software-defned networks: a combined scheduler-per-node approach	2019	Two-segment scheduler model to improve network-wide congestion awareness following	By applying network calculus, the model predicts key performance metrics such as throughput and delay. It utilizes arrival and service curves to accurately represent	The proposed model increases energy consumption due to the complex mathematical model especially when the network size increases.

				analytical approach.	network flows, with a focus on optimizing active queues at individual network nodes.	
7	Steven and Kwan-Yee	A Traffic Meter Based on a Multicolor Marker for Bandwidth Guarantee and Priority Differentiation in SDN Virtual Networks	2019	Multi-Color Marker (MCM) for bandwidth guarantees in SDN virtual networks.	Operates as Remaining Bandwidth Sharing (RBS) mode where unused network capacity is distributed among the virtual networks, allowing them to exceed their guaranteed bandwidth.	It does not address intra-VN packet prioritization. The achievable bandwidth for high-priority traffics is dependent of the VN's guaranteed bandwidth.
8	Lucas and Lasaro	Bandwidth throttling in a P4 switch	2019	P4-based bandwidth throttling mechanism,	Extends the BMv2 switch's capabilities by providing methods accessible through the P414 action profile construct.	It is less adaptive with other switches. Utilizes the P414 version, which has since been updated to P416, with inclusion of several enhancements and new features.
9	Al Breiki et al.	Design and Validation of a Meter Band Rate in OpenFlow and OpenDaylight for Optimizing QoS	2020	Meter Band Rate Evaluation (MBE) mechanism in OpenFlow and OpenDaylight.	Enhances the native QoS capabilities of OpenFlow and OpenDaylight. Allowing dynamic adjustment of traffic rates to be specified relative to other network traffic.	Relies on predefined rates at BRDL, making it difficult to maintain as network size and complexity increases. The approach requires changes to existing OpenFlow components and integration of new elements (BRDL, BRE), making it less adaptable to various protocols without further modifications.

1.2 Gaps in Literature

While prior research has significantly advanced QoS and congestion control in SDN, most existing solutions are tightly coupled with specific SDN protocols such as OpenFlow, resulting in static rate-limiting mechanisms that lack adaptability to dynamic network conditions. Techniques like Meter Band Rate Evaluation (MBE) provide structured rate control but remain constrained by their protocol-dependent nature and limited flexibility in bandwidth management. Notably, none of the existing studies explore dynamic meter-based rate limiting using the P4 language, a protocol-independent data plane programming language. This omission leaves a critical gap in supporting real-time, granular, and adaptive rate control for multimedia traffic in SDN. The proposed Dynamic Meter-Based Rate Limiter (DMB-RL) using P4 addresses this gap by introducing real-time threshold adjustments, adaptive packet sampling, and traffic-aware recirculation, offering a novel path toward flexible and protocol-independent QoS enforcement in modern programmable networks.

1.0. Methodology

2.1 Experimental setup

The experimental setup employed a combination of tools and technologies to create a controlled and replicable testing environment. The key components of the setup include the Mininet Network Emulation Environment to emulate a realistic network. To improve realism, we incorporated publicly available multimedia traffic traces, such as video streaming logs and VoIP patterns, which were replayed in the testbed using iPerf traffic generation tools. This allowed us to observe DMB-RL's behavior under realistic bursty and latency-sensitive conditions, ensuring a more practical evaluation beyond synthetic packet generation. The P4 Network Programming Language was used to program the behavior of the network switches. This included the implementation of custom packet processing pipelines and the dynamic allocation of network resources based on real-time traffic conditions.

Python scripts were developed to automate the configuration of the Mininet environment, control the P4 switches, and collect performance data. These scripts facilitated the systematic execution of experiments and the accurate measurement of key performance metrics such as throughput, latency, Jitter and packet loss.

2.2 Proposed Model

The main logic for the dynamic meter-based rate limiter is written in the form of bandwidth consumption rate to available bandwidth rate using P4 and Python language in between switch links. The P4 compiler compiles the Type of Service "ToS" field set of match-action tables. The proposed solution has utilized an indirect meter in the P4 data structure which is referenced in the action set of the match-action table at the data plane as shown in Figure 1.

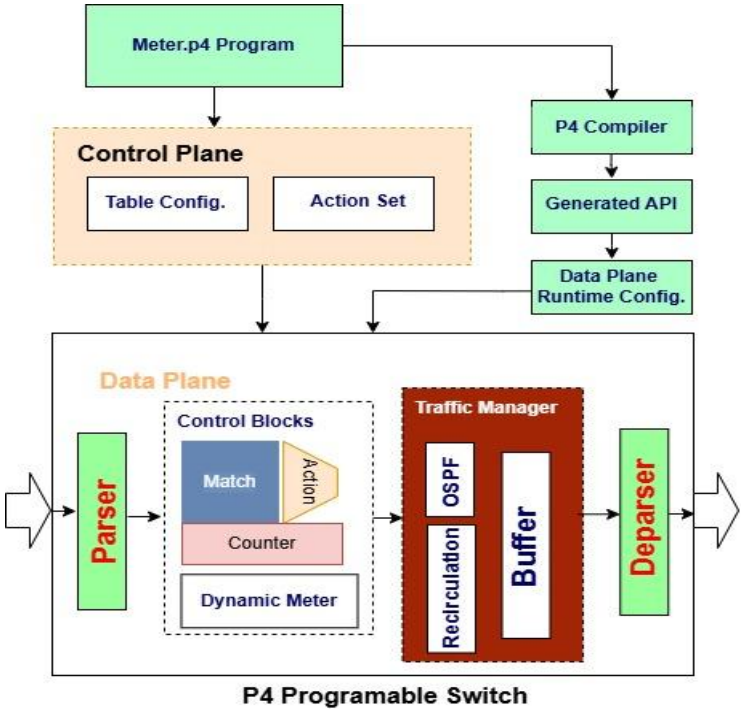

Figure 1: The proposed model's architecture

2.3 Dynamic Threshold for Meter

A meter regulates packet delivery rate within a flow, controlling the overall bandwidth usage [15]. When network conditions change, the meter rate can dynamically adjust, either increasing or decreasing to adapt. Using the P4 Match and action table, packets are prioritized based on their Type of Service (ToS) values, with the priority influencing the threshold set for the indirect meter.

The dynamic meter-based rate limiter leverages the indirect meter option in P4 to efficiently manage bandwidth allocation, as shown in Figure 1. A key innovation in this solution is the inclusion of a link capacity evaluator, which dynamically computes transmission thresholds for the next second. This dynamic calculation addresses issues associated with static bandwidth reservation, such as starvation of non-prioritized traffic and over-provisioning.

Unlike traditional methods that rely on packet sampling, this approach evaluates network state on a per-packet basis, this granularity allows for more precise bandwidth allocation, ensuring that resources are optimally distributed according to current network demands.

The dynamic threshold is determined by real-time network metrics. Let $C(t)$ represent the current link capacity at time t, and $\Lambda(t)$ denote the aggregate packet arrival rate. The threshold rate $R(t+1)$ for the next second is computed as in Equation 1:

$$R(t+1) = min(\max(\Lambda(t), R_{min}), R_{max}) \quad (1)$$

Here, R_{min} and R_{max} are the minimum and maximum rates derived from multimedia traffic requirements.

Measuring the bandwidth consumption involves tracking the amount of data transferred over a network link within a given time interval. This measurement is typically done using counters that record the number of bytes transmitted. Bandwidth consumption can be defined as the process of using a network's transmission capacity to deliver data packets from a source computer to a destination computer. The formula for calculating the consumed bandwidth (B_w) is given by Equation 2:

$$B_w = \frac{(C_{curr} - C_{prev})\ X\ 8}{T_s} \quad (2)$$

Where:

- C_{curr} is the current counter value representing the total bytes transmitted at the end of the sampling interval.
- C_{prev} is the previous counter value at the beginning of the sampling interval.
- 8 converts bytes to bits.
- T_s is the sampling interval in seconds.

This formula provides the consumed bandwidth in bits per second (bps), giving a precise measure of the data transmission rate during the interval. The dynamic threshold setting mechanism adjusts the rate at which packets are delivered based on the calculated real-time bandwidth consumption data.

Dynamic meter-based rate limiter Algorithm

```
metadata {
        bit<32> remaining_capacity;
bit<32> used_bandwidth; }
counter bytes_counter {
        type: bytes;
        size: 32;
direct: true; }
action measure_and_update_metrics() {
        static bit<32> prev_counter_value = 0;
        bit<32> curr_counter_value = bytes_counter.read();
        bit<32> s = <sampling_interval>;
        used_bandwidth = ((curr_counter_value - prev_counter_value) * 8) / s;
        remaining_capacity = read_link_capacity();
        prev_counter_value = curr_counter_value;
        if (used_bandwidth > load_variation_threshold) {
        metadata.sampling_interval = max(metadata.sampling_interval / 2, min_sampling_interval); }
        else if (used_bandwidth < stable_load_threshold) {
        metadata.sampling_interval = min(metadata.sampling_interval * 2, max_sampling_interval); }
}
action set_rate_threshold() {
        const bit<32> R_min = <min_rate>;
        const bit<32> R_max = <max_rate>;
        bit<32> rate_threshold = min(max(used_bandwidth, R_min), R_max);
        rate_threshold = min(rate_threshold, remaining_capacity);
        meter.execute_meter(meter_idx, rate_threshold);
}
control ingress {
        apply {
        measure_and_update_metrics();
        set_rate_threshold(); }
}
control main {
        apply {
        ingress.apply();}
}
```

2.4 Sampling Interval

The sampling interval determines how frequently the network state is assessed and the bandwidth consumption is measured. Selecting the right interval is crucial for balancing accuracy and overhead, as a too-short interval may lead to excessive processing, while a too-long interval might miss rapid fluctuations in network load. Our approach addresses this challenge by optimizing the sampling interval to ensure accurate bandwidth consumption measurement without overwhelming network resource. This work dynamically adjusts the sampling interval based on network conditions to

ensure that unnecessary processing overhead is not incurred. The mechanism used adjusts the sampling interval based on current network conditions. During periods of high network load or rapid fluctuations, the sampling interval is shortened. When the network load is stable, the sampling interval is lengthened. This reduces the processing overhead, as frequent sampling is unnecessary during periods of steady state.
The sampling interval computed with the mathematical model in equation 3:

$$S_{t+1} = \begin{cases} \frac{S_t}{2}, & if \lambda_t > \theta_{high} \\ 2S_{t,} & if \lambda_t < \theta_{low} \\ S_{t,} & otherwise \end{cases} \quad (3)$$

where θ_{high} and θ_{low} define traffic load thresholds.

2.5. Experiment

Real-time bandwidth consumption was captured every second between host_2 and switch_1 in Mininet (Figure 2). This data was recorded using a Python script to calculate available bandwidth for the next second. This approach aligns with recommendations by [16] for maintaining video quality across varying network conditions. Figure 3 displays the dynamic meter rates, demonstrating how the system adapts to real-time traffic demands the experimental test was carried out within the specified ranges to align with the range of bandwidth observed in the work of [14], based on the simulated link capacity of 10mbps, a burst size of 1mbps was allowed to accommodate variability in video traffic. This approach ensures that video quality remains within acceptable limits, even during peak usage, by dynamically adjusting the threshold based on real-time network conditions while prioritizing video traffic.

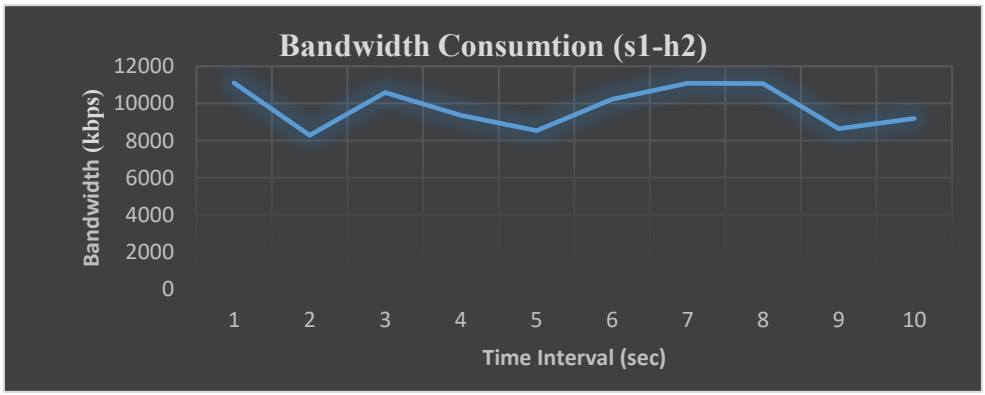

Figure 2: Bandwidth Consumption Between s1-h2

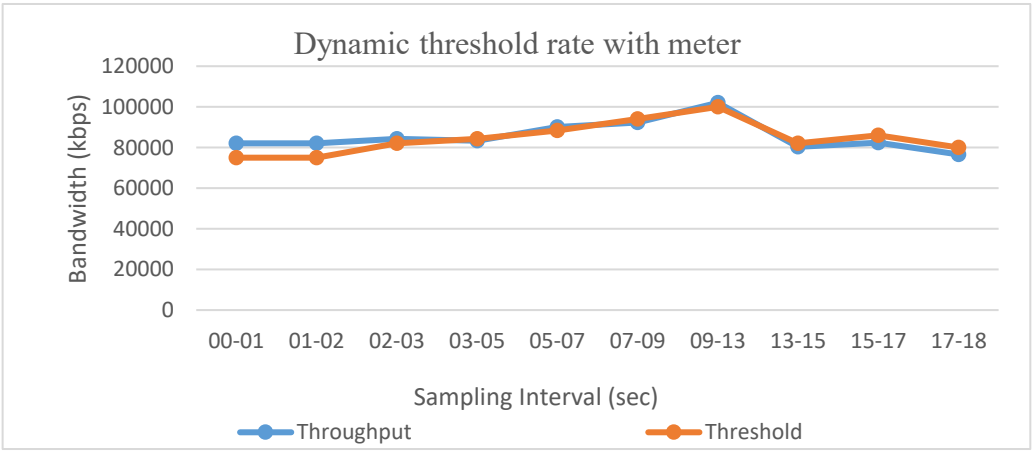

Figure 3: Dynamic threshold rate with meter

3.0. Results and Discussion

This section presents the analysis and discussion of the results obtained from the comparison of Meter Band Rate Evaluator (MBE), and the proposed Dynamic Meter-Based Rate Limiter. The evaluation focuses on four key Quality of Service (QoS) parameters: Packet Loss Rate (PLR), Jitter, Throughput, and Delay. The analysis highlights the performance improvements achieved by the Dynamic Meter in a Software Defined Networking (SDN) environment using our designed algorithm based on P4 language.

3.1 Performance Comparison Based on Packet Loss Rate

The Packet Loss Rate (PLR) over time for the MBE and Dynamic Meter is depicted in Figure 4.

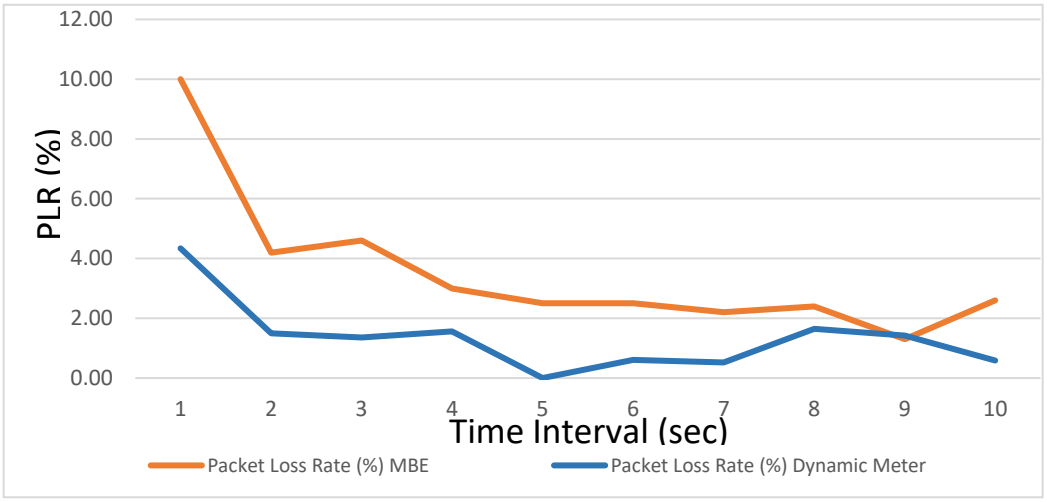

Figure 4: Packet Loss Rate vs Time Interval Graph

The PLR within a transmission interval can be computed using Equation 4:

$$\text{PLR} = \frac{\text{Number of Packets Lost}}{\text{Total Packets Sent}} X \frac{100}{1} \qquad (4)$$

The MBE showed significant improvement in the average PLR but the fluctuations remain noticeable and may not be acceptable for critical application demands like real time multimedia applications which is the subject of this research. On the other hand, the proposed dynamic meter-based rate limiter demonstrated a marked improvement of 61.42% compared with the MBE approach, achieving lower and most stable PLR, ranging from 0.00% to 4.30%. This stability is due to the Dynamic Meter's ability to adjust its measurement intervals and bandwidth threshold for prioritized traffic thereby maintaining a consistent performance even under varying network conditions. Such low and stable PLR is critical for ensuring the smooth delivery of real-time data streams, significantly reducing the chances of interruptions.

3.2 Performance Comparison Based on Jitter

Jitter is calculated with Equation 5:

$$J = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(D_i - \overline{D})^2} \qquad \textbf{(5)}$$

The graph of the Jitter values for each meter type is presented in Figure 5.

Although, the MBE showed better performance when compared with the static meter. This improvement in jitter reflects its enhanced ability to handle network traffic by the use of the Band Rate Description Language (BRDL) and the Band Rate Evaluator (BRE) which introduces some level of dynamism by having different classes of bandwidth and evaluating against current demand in other to decide whether to pass or drop a traffic. However, it will be noted that the proposed Dynamic Meter outperformed the MBE, achieving lower average jitter value of 2038.39ms² or 46.93% improvement which demonstrates the Dynamic Meter's superior capability to maintain stable packet arrival times, this is essential for minimizing disruptions and ensuring a high-quality user experience in real-time applications. The high performance recorded by the Dynamic meter in terms of the Jitter value is due to the fact that the approach of individual packet handling differs from the other two approaches. Rather than sampling packets for threshold determination, individual packet is considered at every other second of the traffic flow to set new meter threshold. This makes the proposed approach very dynamic and network situation aware at all time.

Given the critical importance of timing in these contexts, the arrival of packets that are either too delayed or too premature can result in disruptive glitches, delays, or even data loss, all of which significantly diminish the quality of user experience [17]. Consequently, the assessment and mitigation of jitter is essential in the optimization of network performance within such environments.

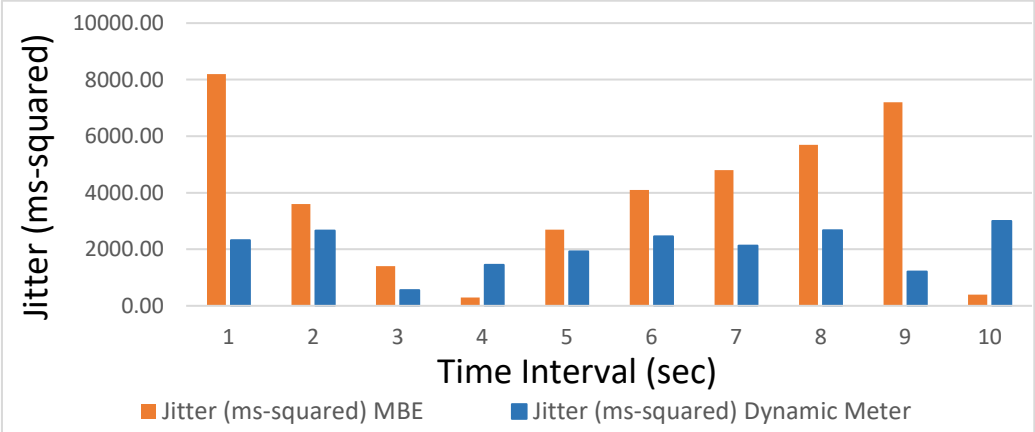

Figure 5: Jitter vs Time Interval Graph

3.3 Performance Comparison Based on Throughput

Equation 6 shows throughput calculation.

$$\text{Throughput} = \frac{\text{Number of Packets Transmitted}}{Time_{(s)}} \qquad (6)$$

Different metering techniques have varying effects on the representation of bandwidth utilization as shown in the graph of Figure 6. The MBE demonstrated improved throughput average at 5,126 (pps) indicating better handling of network traffic compared to the Static methods. The Dynamic Meter consistently achieved the highest throughput, with an average value of 6,673.8 (pps). This indicates how well it can dynamically optimize bandwidth usage through the process of adjusting measurement interval and setting rates as network situation demands ensuring that the network can handle higher data volumes more efficiently while avoiding strict bandwidth reservation that can lead to over-provisioning and under-provisioning as observed in MBE. It should be noted that even if the link capacity is high, it is still very crucial to achieve high throughput for maximizing network efficiency and ensuring that sufficient usable bandwidth is available for high-demand applications like multimedia traffics.

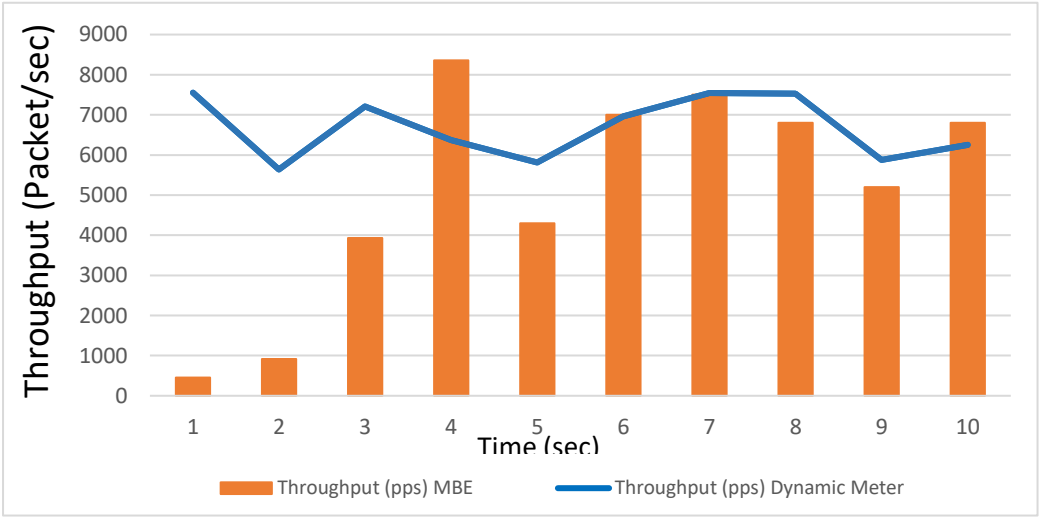

Figure 6: Throughput vs Time Interval Graph

3.4 Performance Comparison Based on Latency

The round-trip time (RTT), which includes the time for a packet to travel to the destination and back is measured as in Equation 7:

$$RTT = \frac{1}{N}\sum_{i=1}^{N}(T_{send,i} - T_{receive,i}) \quad (7)$$

Where:

- RTT is the average round-trip time.
- N is the number of packets or measurements.
- $T_{send,i}$ is the time when the $i-th$ packet was sent.
- $T_{receive,i}$ is the time when the $i-th$ packet was received back.

Figure 7 shows the Delay measurements for each meter type. The Native OpenFlow Static Meter experiences very high latency indicating inefficient traffic management due to the passive nature of the meter and traffic processing at every transmission interval. The MBE showed the least delays with an average of 212.50ms. This can be attributed to the overall consideration given to different traffic classes and bands using the Band Rate Definition Language (BRDL) and Band Rate Evaluator (BRE) of the MBE approach. Each of the traffic class is put into the computation for bandwidth assignment. Therefore, even non-prioritized traffic counts in the decision to pass or drop a traffic in the link. However, this is the only metric where the dynamic meter as proposed in this work performed below the MBE approach. This performance limitation is understandable as the computation overhead of both the dynamic adjustment to the measurement interval and situation aware threshold rate limiting introduces some delay in packet processing time. It is important to note that this latency increase is relatively minor and occurs only in specific traffic patterns, particularly under high burstiness or congestion. The trade-off between slightly increased latency and the significant improvements in throughput (+30.14%) and packet loss reduction (-61.42%) highlights the overall effectiveness of the approach. A solution to improving the dynamic meter's performance in the latency measurement is the introduction of multiple meter data structure suggested as a future research direction. The proposed dynamic rate limiting technique introduces a slight increase in latency of -6.42% performance compared to MBE.

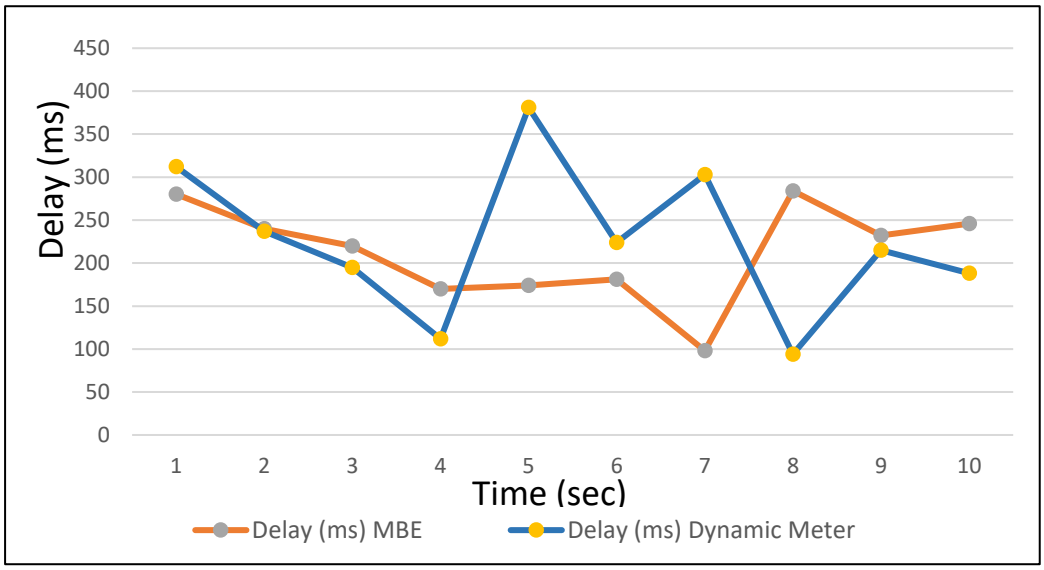

Figure 7: Delay vs Time Interval Graph

Table 2 summarizes the averages of the key performance metrics, including throughput, packet loss, latency, and jitter across the distinct rate limiting scenarios: The improvement of the dynamic meter (implemented in this work) to the MBE (as proposed by [14]) is presented below. The outcomes of this work clearly show the relative effectiveness and efficiency of the dynamic rate limiting in SDN environments using the proposed algorithm built on P4 network language.

Table 2: Summary of performance metrics

Measurement Metrics	MBE Metering	Dynamic Metering	Improvement (Dynamic vs. MBE)
Packet Loss Rate (%)	3.53	1.36	61.42%
Throughput (pps)	5126	6673.8	30.14%
Jitter (ms²)	3839.80	2038.39	46.93%
Latency (ms)	212.5	226.1	-6.42%

The comparative analysis of the MBE of [14] and the proposed Dynamic Meter-Based Rate Limiter reveals significant performance improvements with the Dynamic Meter. Across three of the evaluated QoS parameters—Packet Loss Rate, Throughput and Jitter except for Latency, the Dynamic Meter consistently outperformed the MBE.
It effectively reduced Packet Loss Rate, minimized Jitter, maximized Throughput, and lowered Latency with only a negligible increase in latency, making it highly suitable for real-time and high-demand network applications. These findings validate the efficacy of the Dynamic Meter in enhancing network performance and ensuring a higher quality of service in SDN environments.
To validate the performance improvements of the proposed DMB-RL algorithm, statistical analysis was conducted on the collected QoS metrics—throughput, jitter, and packet loss. Each experiment was run multiple times under identical conditions to ensure repeatability. The results were analyzed using confidence intervals (95%) and two-tailed t-tests to determine statistical significance. The improvements observed in throughput (+30.14%), jitter (-46.93%), and packet loss (-61.42%) were all found to be statistically significant ($p < 0.05$) when compared to the baseline MBE method. These findings confirm that the observed enhancements are not due to random variation but reflect the consistent effectiveness of the DMB-RL algorithm.
The proposed algorithm is written in P4, a protocol-independent language that is supported by modern programmable switch hardware, such as Intel Barefoot Tofino, NetFPGA, and Edgecore switches. These switches allow for line-rate execution of P4 programs, making real-time rate limiting and traffic adaptation not only feasible but also efficient in production-grade networks. The algorithm is modular and vendor-agnostic, allowing it to be embedded into existing SDN-based QoS frameworks with minimal disruption. It can co-exist with traditional OpenFlow-based components, particularly in hybrid networks transitioning to fully programmable data planes.

4.0. Conclusion

In this study, a method that does threshold rate limiting dynamically through real-time monitoring of the link condition and adjusting this measurement interval dynamically based on link performance is proposed. This allows effective use of the network bandwidth when forwarding the packets and also help to regulate and control network traffic in real-time. The dynamic meter has been integrated with traffic prioritization mechanism that is based on the Type of Service (ToS) value. Deploying our methodology in a complex topology, demonstrates the efficiency of the proposed approach. Dynamically changing the meter threshold may still lead to packet loss and increased Latency, especially if the network conditions change very rapidly. Additional research direction for the future is to introduce multiple meter structures for QoS which will mean higher scalability.

References

[1] Pakzad, F., Portmann, M., Tan, W., & Indulska, J. (2015). Efficient Topology Discovery in OpenFlow-based Software Defined Networks. Computer Communications. https://doi.org/10.1016/j.comcom.2015.09.013
[2] Jammal, M., Singh, T., Shami, A., & Asal, R. et al. (2014). Software defined networking: State of the art and research challenges. Computer Networks, 72, 74–98. https://doi.org/10.1016/j.comnet.2014.07.004.
[3] Shaghaghi, A., Kaafar, M. A., Buyya, R., & Jha, S. (2020). Software-Defined Network (SDN) data plane security: Issues, solutions, and future directions. In Handbook of Computer Networks and Cyber Security (pp. 341–387). https://doi.org/10.1007/978-3-030-22277-2_14
[4] Kreutz, D., Ramos, F., Verissimo, P., & Rothenberg, C. E. et al. (2015). Software-Defined Networking: A Comprehensive Survey. In ArXiv e-prints (2014). https://doi.org/10.1109/JPROC.2014.2371999
[5] Xue, L., Ma, X., Luo, X., & Chan, E. W. W. et al. (2018). LinkScope: Towards Detecting Target Link Flooding Attacks. IEEE Transactions on Information Forensics and Security (TIFS). https://doi.org/10.1109/TIFS.2018.2815555

[6] Yan-Wei, C., Yen, L.-H., Wang, W.-C., Chuang, C.-A., Liu, Y.-S., & Tseng, C.-C. (2019). P4-enabled bandwidth management. In Proceedings of the 20th Asia-Pacific Network Operations and Management Symposium (APNOMS). IEICE. https://doi.org/10.23919/APNOMS.2019.8892909

[7] Minh-Thuyen, T., Huynh, T., Hasegawa, M., & Hwang, W.-J. (2016). A rate allocation framework for multi-class services in software-defined networks. Journal of Network and Systems Management, 24(3), 761–783. https://doi.org/10.1007/s10922-016-9368-x

[8] Lin, T.-N., Hsu, Y.-M., Kao, S.-Y., & Chi, P.-W. (2016). OpenE2EQoS: Meter-based method for end-to-end QoS of multimedia services over SDN. IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC) (pp. 1–6). https://doi.org/10.1109/PIMRC.2016.7794972

[9] Li, X., Yan, J., & Ren, H. (2017). Software defined traffic engineering for improving quality of service. China Communications (Volume: 14, Issue: 10, October 2017). Page(s): 12 – 25 https://doi.org/10.1109/CC.2017.8107629

[10] Al-Jawad, A., Shah, P., Gemikonakli, O., & Trestian, R. (2019). Policy-based QoS management framework for software-defined networks. In *Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. https://doi.org/10.1109/NFV-SDN.2018.00039

[11] Darmani, Y., & Sangelaji, M. (2019). QoS-enabled TCP for software-defined networks: A combined scheduler-per-node approach. The Journal of Supercomputing, 75(6), 3456–3476. https://doi.org/10.1007/s11227-019-02837-2

[12] Steven, W. L., & Chan, K.-Y. (2019). A traffic meter based on a multicolor marker for bandwidth guarantee and priority differentiation in SDN virtual networks. IEEE Transactions on Network and Service Management. Advance online publication. https://doi.org/10.1109/TNSM.2019.2923110

[13] Fernandes, L. B., & Camargos, L. (2020). Bandwidth throttling in a P4 switch. In 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) (pp. 91–94). https://doi.org/10.1109/NFV-SDN50289.2020.9289836

[14] Al Breiki, M. S., Zhou, S., & Luo, Y. R. (2020). Design and validation of a meter band rate in OpenFlow and OpenDaylight for optimizing QoS. *Advances in Science, Technology and Engineering Systems Journal*, 5(2), 35–43. https://dx.doi.org/10.25046/aj050205

[15] Lee, S. S. W., & Chan, K.-Y. (2019). A traffic meter based on a multicolor marker for bandwidth guarantee and priority differentiation in SDN virtual networks. IEEE Transactions on Network and Service Management, 16(3), 1046–1058. https://doi.org/10.1109/TNSM.2019.2923110

[16] Uhrina, M., Holesova, A., Sevcik, L., & Bienik, J. (2021). Impact of Scene Content on High Resolution Video Quality. Sensors, 21, 2872. https://doi.org/10.3390/s21082872

[17] Briscoe, B., Brunstrom, A., Petlund, A., Hayes, D., Ros, D., Tsang, I.-J., Gjessing, S., Fairhurst, G., Griwodz, C., & Welzl, M. (2014). Reducing internet latency: A survey of techniques and their merits. *IEEE Communications Surveys & Tutorials*, 18(1), 1–26. https://doi.org/10.1109/COMST.2014.2375213