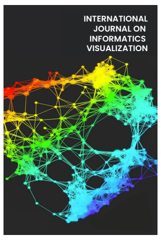
INTERNATIONAL JOURNAL
ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv



Performance Assessment of QoS in Software Defined Networking using Meter Table and Floodlight Controller

Diyar Jamal Hamad [a,b,*], Khirota Gorgees Yalda [a,b], Nicolae Țăpuș [a]

[a] *Computer Science, Politehnica University of Bucharest, Splaiul Independenței 313, București 060042, Bucharest, Romania*
[b] *IT, Erbil Polytechnic University, 42VP+5R2, 120 m street, Erbil, Iraq*
*Corresponding author: *diyar.hamad@epu.edu.iq*

***Abstract*— The quality of service is not the same in all parts of the network. Some areas experience a low level and others a higher level of fixed quality services. The shortcomings in legacy networks encouraged researchers to find a new paradigm of the network to obviate legacy networks' deficiencies. The effort to create network services is called Quality of Service (QoS). Software-Defined Networking (SDN) focuses on separating the control layer from the data layer, and their communication is done through a central controller named SDN controller. After separation, the data layer moves the packets through the network according to the commands it receives from the controller. The controller obtains applications (QoS requests), translates them to low-level instructions, and implements them in the data layer. In this paper, we create an infrastructure for Quality of Service (QoS) in tree topology using a meter table per flow in Software Defined Networking Floodlight open-source controller. Meters are introduced into the OpenFlow protocol version 1.3, which calculates the packet rates allocated to them and allows control of those packet rates. Meters are directly connected to flow entry. Any flow entry can determine a meter in its command collection, which calculates and supervises the sum of all flow entries to which it is connected. When we get statistics from the meter table in each switch, we manage the network and affect the routing algorithms.**

***Keywords*— SDN; QoS; floodlight; mininet; Open vSwitch.**

Manuscript received 17 Oct. 2022; revised 11 Mar. 2023; accepted 2 Apr. 2023. Date of publication 10 Sep. 2023.

I. Introduction

It should be considered that the Internet plays a big role in entrepreneurship, so it is impossible to ignore the deficient service quality it provides. The fast broadening of the Internet and the rise in traffic levels make it a little difficult for users to benefit from stable service quality [1]. Users inquire about a network to support various service levels, unlike today's Internet, which works based on a single best-effort service level [2]. End-to-end QoS on the Internet can be broken down into a number of categories, such as End-user QoS, ISP network QoS, and Internet backbone QoS. The features of QoS issues vary based on the needs they must satisfy. Specialized technologies are used to resolve each of these requirements. For instance, bandwidth and traffic engineering issues largely contribute to the QoS challenges Internet backbones face. QoS will not be an issue if we can supply enough Bandwidth in the backbone. Nevertheless, this area has traffic congestion because end-user connections are concentrated within the ISP network [3]. Traffic policing combined with QoS in reducing network congestion is effective [4]. In addition to clearing up network congestion, QoS also helps to keep the LAN stable [5]. Service providers and sellers of network devices compete by providing higher quality service for various kinds of media like data, audio, and video to acquire customers. The authors described a framework to assist QoS equipping for reserving the resource demands of various kinds of data flows in reference [6]. It is feasible that there would not be sufficient accessible routes to redirect QoS flows, which results in deficient QoS flow transfer performance in complicated networks [7]. However, providing a high QoS level is difficult because of the many restrictions applied in the legacy networks. For instance, their complexity makes legacy networks difficult to maintain and manage. In addition, they cannot adapt to changes and new QoS demands. Such weaknesses are mostly created in legacy networks because of the nature in which both data and control layers are combined and integrated into network equipment [8]. Software Defined Networking (SDN) is a network model, and its most important attribute is offering centralized control and an overall network vision. It is difficult to control legacy networks because they are complex and complicated. Several reasons exist, including vertical integration of data and control planes and manufacturer-specific data and control planes. By decoupling the Control plane from the Data plane, SDN provided a solution to these long-standing issues, allowing the routing network to be more flexible and the control plane to be more centralized [9]. There is no programmability feature in legacy networks, but SDN provides this feature. The control layer resides inside the servers and remotely manages the forwarding layer, which resides in the network and can offer fundamental transport services like scheduling, metering, and classification. The centralized specification of SDN reduces the burden on traditional transmitting devices since the controller is responsible for partial algorithms and protocols here. A summary of the results of the performance of the various controllers depending on specific QoS factors, such as scalability, reliability, load balancing, and consistency, is provided in reference [10]. The SDN controller performance tests are conducted using the criteria (such as Bandwidth) [7]. OpenFlow protocol is the technology and interaction channel to actualize SDN, providing interaction between the forwarding and control layers [11]. OpenFlow protocol, which is the real criterion of SDN, regarded QoS as a component of its operations from the release of its first version. In its first version, OpenFlow introduces an enqueue technique by which specific traffic can be routed to a specific queue that preserves QoS. An OpenFlow architecture that helps microflow-based QoS extensibility is offered in references [12], [13]. There is also an innovative framework built on Software-defined Networking for balancing traffic between IoT servers and meeting the QoS requirements of various IoT services [14]. To build the queue, SDN offers OVSDB and OF-Config protocols. The OVSDB protocol was previously implemented in Open vSwitch, an OpenFlow switch software. The newest versions of OpenFlow, which is 1.4 offer per-flow meters, which offer uncomplicated functioning like straight control of packet rate per flow-entry, which means a limited rate, whereas the queues are connected to the ports to which traffic is dispatched [15]. QoS is a topic that many researchers have investigated. Before and after the SDN's introduction, research was conducted on the issue of running QoS in networks [16]. QoS is configured using QoS policy-maps to highlight the advantages of utilizing SDN technology for supplying QoS provisioning techniques [17]. Two scenarios are used to test this configuration: a software-defined network and a legacy IP network. The study [18] implements an SDN architecture consisting of a Floodlight controller with a switching component, one OpenFlow switch, and several nodes using a Mininet emulator, based on new OpenFlow functionality, such as meters. We obtained that the quality of service in an SDN environment using Meters is better than any other way explained in the previous works. In this paper We propose a resource management model for SDN. Floodlight Controller [19] is used to build our framework, and Mininet 2.3 [20] is used to imitate the experiment's tree topology. The forwarding device is a multilayer virtual switch called Open vSwitch 2.13.8 [21]. OpenFlow1.3 serves as the conduit for communication between the controller and the transmitter. Our goal is to increase and improve the QoS.

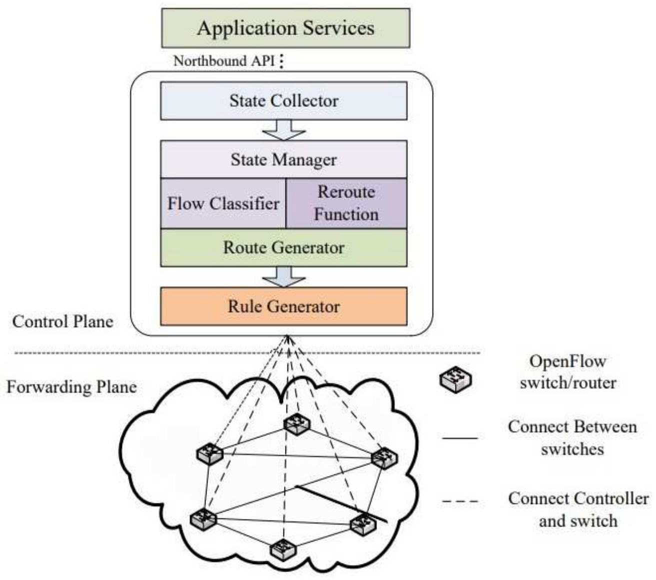


Fig. 1 SDN Architecture [9]

II. Materials And Method

Quality of Service (QoS) has been defined in several ways. One definition says: it is the ability to control traffic in the network so that the network satisfies the requirements of users and applications [22] and all network parts meet certain certainty that their needs will be fulfilled [3]. Through QoS some network attributes, such as error and transport ratio, can be evaluated, enhanced, and even partially guaranteed [11]. QoS networks should have a method to observe the distribution of resources among users and applications. The emergence of QoS answered recent requests from novel applications, particularly multimedia real-time applications [23]. The existence of these applications created limits for the allowable time delay of transferring data through a network. We should consider that; however, the issue of QoS became important not long ago, yet experts previously predicted the idea of QoS. In the primary designation of IP, a byte named Type of Service (ToS) resides in the IP header, which is reserved to clear the way for the QoS. Up to the late eighties, most IP implementations neglected this byte because there wasn't awareness of QoS importance yet [23]. In this paper, we conducted this experiment to demonstrate how our QoS control technique impacts the effectiveness of QoS per meter. We emulated a real-world network using an imitated test tool (iperf) [24]. All switches add flow entry values to generate QoS with a meter table. A UDP link generated 1Gbps of traffic. In the simulated network, 0% of packets are dropped.

A. *QoS Frameworks*

The Internet, one of the world's biggest WANs, is going through large alterations, confronting sudden challenges to fulfill different service requirements. Applications like video conferencing have gained many fans on the Internet, such applications also demand the use of resources with higher Bandwidth. Today's internet structure cannot provide good services, so we need solutions to reach the desired quality of services [11]. Adaptive QoS-based routing and resource reservation (VQoSRR) for video streaming enables SDN networks to handle video demands and improve user experience over best-effort networks introduced in reference [25].

IntServ and DiffServ are the classical QoS frameworks in today's Internet [22]. There is also MPLS, but it is more of a traffic engineering protocol than QoS framework [3].

1) Integrated Services: Resource reservation occurs on a per-flow basis for all network devices [15]. Applications with Real-time features must reserve resources and set routes prior to transferring data; RSVP (Resource Reservation Setup Protocol) is utilized to achieve this objective. The transmitter transmits a PATH message to the recipient to determine the traffic status. Every midway router across the route sends the PATH message to the subsequent node specified by the routing algorithm. After the recipient receives the PATH message, it responds with a RESV message to request the required flow resources. Every midway router across the route can accept or refuse the demand included in the RESV message. If the request is denied, the router sends an error message to the recipient, and the signaling procedure will be halted. But if the request is granted, the required flow resources will be set aside, and the router will be updated with the necessary flow status data [3]. The network can offer QoS with IntServ, but because it is based on per-flow, it has scalability issues [11][26].

2) Differentiated Services: At the network ingress routers (network boundary), traffic in DiffServ is categorized and annotated in DSCP (DiffServ code point, a field in IP packets' header) to form various classes of packets. The IPv4 TOS octet or the IPv6 traffic class octet is where DSCP is stored. Various categories of packets get dissimilar services, enabling priority for packet discard and delay indicators. As a result, IP packets can be classified into various categories of QoS, so each is managed in a separate queue of network components like switches or routers [3]. DiffServ introduces a bandwidth broker as part of its policy management module, network, and logical resource. There is a bandwidth broker in every Diffserv independent system. The bandwidth broker upholds regulations and bargains SLAs (service-level agreements) with clients and other adjacent bandwidth broker domains [15]. The most important benefit of DiffServ is to facilitate network operations. This helps the DiffServ network easily collaborate with other networks and makes resource assignments more flexible [3]. Despite the positive points, DiffServ cannot achieve the end-to-end connection without source reservation [11]. Because of a deficiency in both QoS frameworks, they are not yet fully adopted in today's networks. While both frameworks are combined in some solutions to use their best features, they reveal the unavoidable issues in large networks [18].

3) MPLS: MPLS (Multi-Protocol Label Switching) is a packet-forwarding protocol based on packet forwarding. Each packet is labeled as it enters an area with MPLS capability. The labels are the foundation for later packet classification, sending, and services. In some ways, MPLS is like DiffServ since both mark the traffic at the entrance borders of a network and unmark it at the exit points. But contrary to DiffServ, which uses marking to specify the precedence in a router, marking in MPLS, which is a 20-bit label, is intended to specify the next router node [3], [27]

B. QoS Performance Measures

QoS is quantitatively a warranty or limit on a specific network's working metrics [28]. The most popular working metrics are reliability, packet loss, packet delay and jitter, and Bandwidth.

1) Reliability: It can be considered as the medium's typical mistake ratio. The reliability rate is lowered by changing the sequence of packets during transfer, sending packets to the recipient in a dissimilar sequence than when they were first transmitted by the sender or even missing packets due to transitory routing loops [1].

2) Packet Loss: If packet loss trespasses a certain threshold, some applications might not work accurately or may not work at all. In some circumstances, this amount perhaps is zero. Accordingly, some applications may call for assurances on the missing packet ratio for QoS to be considering [29]. When the quantity of arriving packets greatly outnumbers the queue's capacity, packet drops at aggregation locations can cause packet loss. On the transmission wire, corrupt packets might potentially result in packet loss [22].

3) Packet Delay and Jitter: Delay, also called latency [22]is the interval from sending a packet to passing through the network and reaching the receiver [1]. It comes in three different varieties: propagation delay, switching delay, and serialization delay. The time it takes for a signal to move from a sender to a recipient is known as the propagation delay. As a result, this kind of delay depends on the link medium, and the distance covered. The time interval between the arrival of a packet and the beginning of retransfer is known as the switching delay. This delay is affected by the speed of the device. The time a device spends synchronizing the packet on a specific output ratio is known as serialization delay, often referred to as transmission delay. Packet size and bandwidth influence this type of delay [30]. The network's overall usability is also impacted by other delays, such as the type of data being transmitted, the volume of traffic, and network conditions. This change in packet latency is known as packet jitter. When the network is busy, queues will accumulate at the routers and impact the latency.

4) Bandwidth: The transfer capacity of an electronic line is referred to as Bandwidth. Theoretically, it represents the scale of the potential forwarding ratio. It represents the pipe size needed for network communication by application software. Bandwidth is important because it specifies the capacity of the channel or the utmost information amount that can be transmitted [31].

C. QoS Levels

The applications that need QoS vary in priority levels and assurance that they ask to perform QoS. Some tasks do not ask for any assurances. On the other hand, some tasks demand definite assurance to be invulnerable. There are many QoS degrees between these two types. Hard QoS, soft QoS, and best-effort service are the three major groups into which these degrees of QoS have been divided [32].

1) Hard QoS: Hard QoS is also known as guaranteed service. It stands for the QoS level for those applications that need complete assurances on the least network resources required for proper operation. Earlier network resource reservation over a route is typically undertaken for the network to offer or refuse the demanded assurance. Multimedia applications that transmit audio and/or video data in real-time are examples of applications that call for Hard QoS [33].

2) Soft QoS: Soft QoS is also called DiffServ. There are no total assurances provided at this QoS level. Instead, various tasks are given varying priorities. Applications are thus divided into various priority classes. When total assurance is not required, this strategy works extremely well for many application flows [34]. For instance, to guarantee the constant availability of no less than the fundamental connectivity and performance, network control traffic might be given superior priority over other data interactions.

3) Best-effort service: The best-effort service offers absolutely no assurance [35]. It can't indeed be referred to as a QoS. Numerous network applications function quite effectively with best-effort service. The File Transfer Protocol (FTP) is an example of these applications. The transfer's success or failure is the only factor that counts. It is crucial to remember that the last level is the only one the Internet can now offer [23][36].

D. QoS Routing

Numerous aspects complicate the QoS routing issue. One is the variety of demands and assurances of several distributed computing applications operating concurrently [37]. The other important concern is the inability to preserve reliable network status information in a sizable, dynamically changing network. Each node in the network must keep track of its local state in order to retain network state information. Global state information can then be created by merging all local states. A node generally uses either the distance-vector algorithm or the link state to preserve the network's global state information. This is accomplished by repeatedly exchanging the local states among all network nodes using the selected algorithm [38].

III. Results and Discussion

The meter table comprises meter entries that characterize per-flow meters [39]. Per-flow meters permit OpenFlow to execute different ordinary QoS operations, like rate-limiting, which can be corporate with per-port queues to execute complicated QoS frameworks, like DiffServ. It is possible to use several meters in a table by a special technique, which uses a separate set of flow entries. Placing them in sequential flow tables allows different meters to be operated on the same set of packets. Every meter entry has a unique meter ID which is recognized by it and includes the following information:

- Meter ID: it is an unsigned integer number that exclusively identifies the meter, and it is 32-bit in size.
- Meter bands: a disorganized record of meter bands where every meter band determines the band rate and packet processing method.
- Counters: when packets are processed by the meter, they are updated.

Our scheme is set up in Floodlight Controller (Master) [19], and Mininet 2.3 [20] is used to simulate the experiment's tree topology in Figure 2. Mininet is the most popular network emulator and simulator. It has advantages in terms of scalability and offers quick simulation speeds by utilizing OS-level virtualization abilities. It functions with many SDN tools and open-source SDN projects and completely supports the OpenFlow protocol. Due to this benefit, most SDN research uses Mininet [40].

In this experiment scenario, the forwarding device is Open vSwitch 2.13.8 [21], a multilayer virtual switch. The Linux Foundation implements the Open vSwitch. The Open vSwitch contains a function called learn. When receiving packets fit the rules, the learn function can create, change, or remove the rules [41]. The communication interface between the controller and the transmitter is OpenFlow1.3. OpenFlow standard 1.3 is the setting for the Open vSwitches. Switches on the network carry out traffic metering. To see how our QoS control strategy affects the performance of QoS per meter, we created a topology with sixteen hosts and turned one of them into a server to observe the data arriving or losing rate when other hosts generate the traffic.

For each switch or router, QoS designs are given to queues connected to ports. Each port is assigned many queues that have been separated into several levels (The level number is approximately 8). Before being sent from a port, packets that are going to be redirected are first allocated to one of these queues. These queues are configurable with various QoS characteristics, such as a minimum and maximum bandwidth cap [11]. Comparatively, a queue processes packets at a predetermined rate, accepting packets for output.

It is important to note that meters and queues are complementary and not different implementations of the same thing. Metrics are not a replacement for queues, as is a common misconception. Queues require out-of-band definition (e.g., using OpenvSwitch), while meters can be installed, modified, and removed at runtime using OpenFlow. Meters can be compared to flows. Meter tables are abstractions defined by OpenFlow, which contain rows of meters. It is possible to manipulate these meters in much the same way that flows are. It is cool to send packets as output and receive packets as input, just like flows do. We added flows for the network switches to set up each switch by adding flows and meters and demonstrating the switch's QoS. Also, a user can give each switch two commands to activate the meter table and start the meter counter. This process is repeated for each switch.

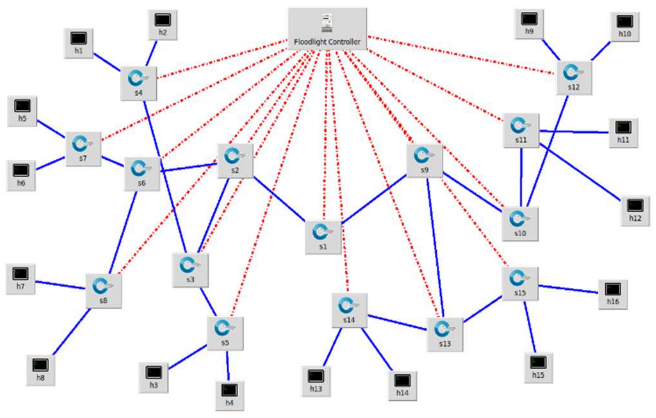


Fig. 2 Network topology

After setting up the meter with the flows for each switch, the Floodlight controller will divide each network with hosts and switches connected to them to a subnetwork, as illustrated in Figure 3. Although the networks appear to have different domains, the controller has permission over all domains. As the figure shows, a switch can only connect to one domain. This indicates that the meter has good network management skills and a reliable characteristic.

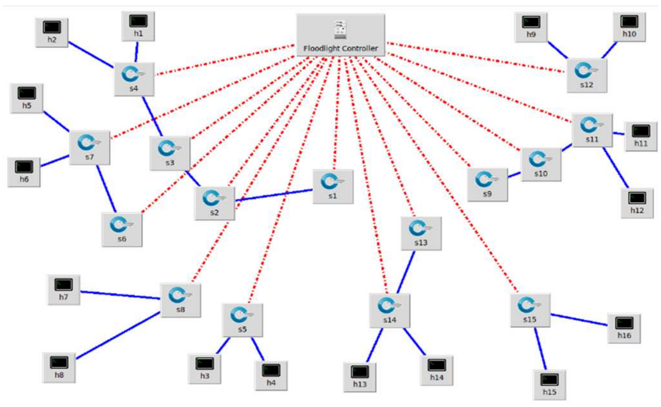


Fig. 3 A subnetwork Topology

We used the iperf network test tool [24] and created the flows for all switches to imitate the traffic found in real-world environments. Installing flow entry values on all switches is the initial step in creating QoS with a meter table. Then, all switches which are connected to hosts are placed with meter entries. The first host receives traffic from other hosts. Table I illustrates the results of sending 1Gbps over a UDP link. 0% of packets from the hosts to the server are dropped.

TABLE I
SENDING 1GBPS DATA FROM HOST 2 TO THE SERVER

Time(s)	Transfer	Bandwidth
0	128 MBytes	1.07Gbits/sec
10	128 MBytes	1.07Gbits/sec
20	128 MBytes	1.07Gbits/sec
30	128 MBytes	1.07Gbits/sec
40	128 Mbytes	1.07Gbits/sec
50	128 Mbytes	1.07Gbits/sec
60	128 MBytes	1.07Gbits/sec

We generate traffic from all other hosts to the server to ensure that the meter's counter on each switch increases. The process repeats for all other hosts, and the result is the same for all of them, as shown in Table I. By meters, traffic can be monitored before being sent out by complementing the queue framework which is currently in place. Meters can provide a more precise measurement of a flow's ingress rate. The first counter in the Meter Table is the packet count, and the second counter is the byte count. Figure 4 illustrates the results after collecting data from the meter table in each switch per second and the byte counter, which increases in the meter table. After the switch's input port generates traffic, the meter counter increases. The information collected from the switch is added to the counter every second. The data collection for each second is calculated for that specific second. Due to the wide Y axis resulting from using the Byte measure, we used the Excel formula to convert Byte to MegaByte.

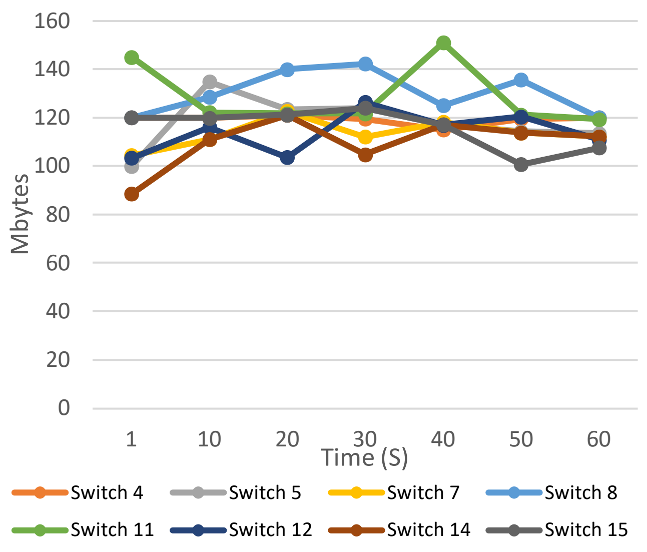


Fig. 4 Byte count in Meter Table

Figure 5 illustrates the time and number of packets arriving at the switch every 10 seconds. Here we can observe that switches are similar in optimization, indicating that they all offer the same level of service with a small quantity of low and height; in this case, we can modify or change the routing algorithm to make it a better and faster network. We performed the same operations using 5GB, and the results were similar to those from 1GB. A better resource management model was demonstrated through two different test cases.

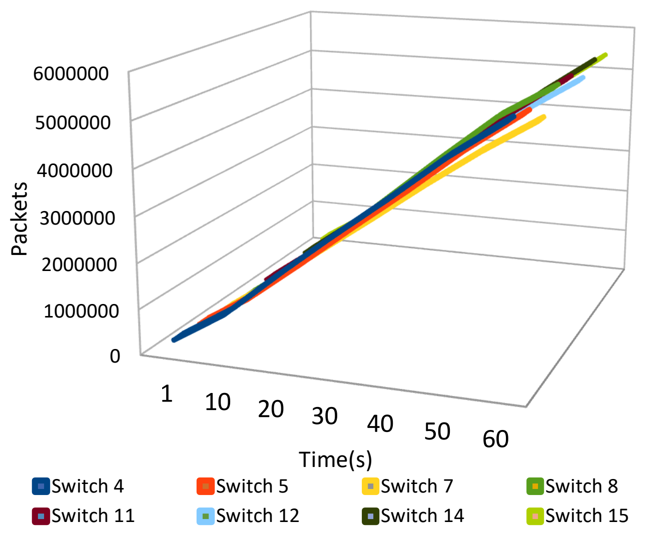


Fig. 5 Packet rate in flow tables

IV. CONCLUSION

This paper uses tree topology and the Floodlight controller to configure QoS in an SDN environment. We obtained the improvement of QoS in our framework, which has been set up in OpenvSwitch. The outcomes are part of our effort to compile statistics from meter and flow tables, which ensures the requirement of QoS flows. We can see the results of our research in Figure 4, which illustrates the monitoring of data transferred between switches that have hosts connected to them; when traffic is created, the counter starts to increase. The counter will not increase and stay in the last value if there is no traffic. Since collecting results every 10 seconds, we can say that meters are one of the most important aspects of SDN network management; by adding meters to the queue architecture, traffic could be watched before being emitted. The ingress rate of a flow can be measured with greater accuracy by meters. In the future, we will administer the SDN network using the meter table and group table together.

REFERENCES

[1] A. H. Sodhro *et al.*, "Quality of Service Optimization in an IoT-Driven Intelligent Transportation System," *IEEE Wirel Commun*, vol. 26, no. 6, pp. 10–17, Dec. 2019, doi: 10.1109/MWC.001.1900085.

[2] R. Al-Haddad and E. S. Velazquez, "A Survey of Quality of Service (QoS) Protocols and Software-Defined Networks (SDN)," 2019, pp. 527–545. doi: 10.1007/978-3-030-01177-2_38.

[3] W. M. H. Azamuddin, R. Hassan, A. H. M. Aman, M. K. Hasan, and A. S. Al-Khaleefa, "Quality of Service (QoS) Management for Local Area Network (LAN) Using Traffic Policy Technique to Secure Congestion," *Computers*, vol. 9, no. 2, p. 39, May 2020, doi: 10.3390/computers9020039.

[4] M. Rezaee and M. H. Yaghmaee Moghaddam, "SDN-Based Quality of Service Networking for Wide Area Measurement System," *IEEE Trans Industr Inform*, vol. 16, no. 5, pp. 3018–3028, May 2020, doi: 10.1109/TII.2019.2893865.

[5] S. K. Keshari, V. Kansal, and S. Kumar, "A Systematic Review of Quality of Services (QoS) in Software Defined Networking (SDN)," *Wirel Pers Commun*, vol. 116, no. 3, pp. 2593–2614, Feb. 2021, doi: 10.1007/s11277-020-07812-2.

[6] A. Montazerolghaem and M. H. Yaghmaee, "Load-Balanced and QoS-Aware Software-Defined Internet of Things," *IEEE Internet Things J*, vol. 7, no. 4, pp. 3323–3337, Apr. 2020, doi: 10.1109/JIOT.2020.2967081.

[7] Q. Tang, H. Zhang, J. Dong, and L. Zhang, "Elephant Flow Detection Mechanism in SDN-Based Data Center Networks," *Sci Program*, vol. 2020, pp. 1–8, Sep. 2020, doi: 10.1155/2020/8888375.

[8] H. E. Egilmez, S. Tahsin Dane, K. T. Bagci, and A. Murat Tekalp, "OpenQoS: An OpenFlow Controller Design for Multimedia Delivery with End-to-End Quality of Service over Software-Defined Networks."

[9] S. H. Haji *et al.*, "Comparison of Software Defined Networking with Traditional Networking," *Asian Journal of Research in Computer Science*, pp. 1–18, May 2021, doi: 10.9734/ajrcos/2021/v9i230216.

[10] B. K. Umrao and D. K. Yadav, "Algorithms for functionalities of virtual network: a survey," *J Supercomput*, vol. 77, no. 7, pp. 7368–7439, Jul. 2021, doi: 10.1007/s11227-020-03502-9.

[11] K. N. Qureshi, R. Hussain, and G. Jeon, "A Distributed Software Defined Networking Model to Improve the Scalability and Quality of Services for Flexible Green Energy Internet for Smart Grid Systems," *Computers & Electrical Engineering*, vol. 84, p. 106634, 2020, doi: 10.1016/j.compeleceng.2020.106634.

[12] M. Afaq, S. U. Rehman, and W.-C. Song, "Visualization of elephant flows and QoS provisioning in SDN-based networks," in *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Aug. 2015, pp. 444–447. doi: 10.1109/APNOMS.2015.7275384.

[13] M. O. Elbasheer, A. Aldegheishem, N. Alrajeh, and J. Lloret, "Video Streaming Adaptive QoS Routing with Resource Reservation (VQoSRR) Model for SDN Networks," *Electronics (Basel)*, vol. 11, no. 8, p. 1252, Apr. 2022, doi: 10.3390/electronics11081252.

[14] A. T. Oliveira, B. J. C. A. Martins, M. F. Moreno, A. B. Vieira, A. T. A. Gomes, and A. Ziviani, "SDN-Based Architecture for Providing QoS to High Performance Distributed Applications," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, Jun. 2018, pp. 00602–00607. doi: 10.1109/ISCC.2018.8538694.

[15] S. Sharma *et al.*, "Implementing Quality of Service for the Software Defined Networking Enabled Future Internet," in *2014 Third European Workshop on Software Defined Networks*, Sep. 2014, pp. 49–54. doi: 10.1109/EWSDN.2014.36.

[16] Y. Bi, G. Han, C. Lin, Y. Peng, H. Pu, and Y. Jia, "Intelligent Quality of Service Aware Traffic Forwarding for Software-Defined Networking/Open Shortest Path First Hybrid Industrial Internet," *IEEE Trans Industr Inform*, vol. 16, no. 2, pp. 1395–1405, Feb. 2020, doi: 10.1109/TII.2019.2946045.

[17] M. U. Younus, S. ul Islam, I. Ali, S. Khan, and M. K. Khan, "A survey on software defined networking enabled smart buildings: Architecture, challenges and use cases," *Journal of Network and Computer Applications*, vol. 137, pp. 62–77, 2019, doi: 10.1016/j.jnca.2019.04.002.

[18] Y. Li, X. Guo, X. Pang, B. Peng, X. Li, and P. Zhang, "Performance Analysis of Floodlight and Ryu SDN Controllers under Mininet Simulator," in *2020 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, Aug. 2020, pp. 85–90. doi: 10.1109/ICCCWorkshops49972.2020.9209935.

[19] S. Asadollahi and B. Goswami, "Experimenting with scalability of floodlight controller in software defined networks," in *2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, Dec. 2017, pp. 288–292. doi: 10.1109/ICEECCOT.2017.8284684.

[20] B. Lantz, A. A. Díaz-Montiel, J. Yu, C. Rios, M. Ruffini, and D. Kilper, "Demonstration of Software-Defined Packet-Optical Network Emulation with Mininet-Optical and ONOS," in *2020 Optical Fiber Communications Conference and Exhibition (OFC)*, 2020, pp. 1–3.

[21] P. Emmerich, D. Raumer, S. Gallenmüller, F. Wohlfart, and G. Carle, "Throughput and Latency of Virtual Switching with Open vSwitch: A Quantitative Analysis," *Journal of Network and Systems Management*, vol. 26, no. 2, pp. 314–338, Apr. 2018, doi: 10.1007/s10922-017-9417-0.

[22] H. Hendrawan, P. Sukarno, and M. A. Nugroho, "Quality of Service (QoS) Comparison Analysis of Snort IDS and Bro IDS Application in Software Define Network (SDN) Architecture," in *2019 7th International Conference on Information and Communication Technology (ICoICT)*, Jul. 2019, pp. 1–7. doi: 10.1109/ICoICT.2019.8835211.

[23] R. Li and P. Zhu, "Spectrum Allocation Strategies Based on QoS in Cognitive Vehicle Networks," *IEEE Access*, vol. 8, pp. 99922–99933, 2020, doi: 10.1109/ACCESS.2020.2997936.

[24] M. T. Naing, T. T. Khaing, and A. H. Maw, "Evaluation of TCP and UDP Traffic over Software-Defined Networking," in *2019 International Conference on Advanced Information Technologies (ICAIT)*, Nov. 2019, pp. 7–12. doi: 10.1109/AITC.2019.8921086.

[25] H. Babbar and S. Rani, "Performance Evaluation of QoS metrics in Software Defined Networking using Ryu Controller," *IOP Conf Ser Mater Sci Eng*, vol. 1022, no. 1, p. 012024, Jan. 2021, doi: 10.1088/1757-899X/1022/1/012024.

[26] Y.-F. Huang, C.-B. Lin, C.-M. Chung, and C.-M. Chen, "Research on QoS Classification of Network Encrypted Traffic Behavior Based on Machine Learning," *Electronics (Basel)*, vol. 10, no. 12, p. 1376, Jun. 2021, doi: 10.3390/electronics10121376.

[27] A. Bahnasse, F. E. Louhab, H. Ait Oulahyane, M. Talea, and A. Bakali, "Novel SDN architecture for smart MPLS Traffic Engineering-DiffServ Aware management," *Future Generation Computer Systems*, vol. 87, pp. 115–126, Oct. 2018, doi: 10.1016/j.future.2018.04.066.

[28] M. S. al Breiki, S. Zhou, and Y. R. Luo, "Development of OpenFlow Native Capabilities to optimize QoS," in *2020 Seventh International Conference on Software Defined Systems (SDS)*, Apr. 2020, pp. 67–74. doi: 10.1109/SDS49854.2020.9143890.

[29] S. Jung, J. Kim, and J.-H. Kim, "Intelligent Active Queue Management for Stabilized QoS Guarantees in 5G Mobile Networks," *IEEE Syst J*, vol. 15, no. 3, pp. 4293–4302, Sep. 2021, doi: 10.1109/JSYST.2020.3014231.

[30] A. Charisma, A. D. Setiawan, G. Megiyanto Rahmatullah, and M. R. Hidayat, "Analysis Quality of Service (QoS) on 4G Telkomsel Networks In Soreang," in *2019 IEEE 13th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, Oct. 2019, pp. 145–148. doi: 10.1109/TSSA48701.2019.8985489.

[31] M. Singh and G. Baranwal, "Quality of Service (QoS) in Internet of Things," in *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, Feb. 2018, pp. 1–6. doi: 10.1109/IoT-SIU.2018.8519862.

[32] B. Sarra, M. Saleh, and H. Saad, "Ensuring QoS and Efficiency of Vehicular Networks by SDVN-IoV," in *2020 International Conference on Advanced Aspects of Software Engineering (ICAASE)*, Nov. 2020, pp. 1–6. doi: 10.1109/ICAASE51408.2020.9380115.

[33] H. Zhong, F. Wu, Y. Xu, and J. Cui, "QoS-Aware Multicast for Scalable Video Streaming in Software-Defined Networks," *IEEE Trans Multimedia*, vol. 23, pp. 982–994, 2021, doi: 10.1109/TMM.2020.2991539.

[34] S. Mollahasani, M. Erol-Kantarci, M. Hirab, H. Dehghan, and R. Wilson, "Actor-Critic Learning Based QoS-Aware Scheduler for Reconfigurable Wireless Networks," *IEEE Trans Netw Sci Eng*, vol. 9, no. 1, pp. 45–54, Jan. 2022, doi: 10.1109/TNSE.2021.3070476.

[35] F. Khan *et al.*, "A Quality of Service-Aware Secured Communication Scheme for Internet of Things-Based Networks," *Sensors*, vol. 19, no. 19, p. 4321, Oct. 2019, doi: 10.3390/s19194321.

[36] G. White, A. Palade, and S. Clarke, "Forecasting QoS Attributes Using LSTM Networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2018, pp. 1–8. doi: 10.1109/IJCNN.2018.8489052.

[37] M. A. Gawas and S. Govekar, "State-of-Art and Open Issues of Cross-Layer Design and QOS Routing in Internet of Vehicles," *Wirel Pers Commun*, vol. 116, no. 3, pp. 2261–2297, Feb. 2021, doi: 10.1007/s11277-020-07790-5.

[38] A. Nauman, Y. A. Qadri, M. Amjad, Y. bin Zikria, M. K. Afzal, and S. W. Kim, "Multimedia Internet of Things: A Comprehensive Survey," *IEEE Access*, vol. 8, pp. 8202–8250, 2020, doi: 10.1109/ACCESS.2020.2964280.

[39] S. D. A. Shah, M. A. Gregory, and S. Li, "Cloud-Native Network Slicing Using Software Defined Networking Based Multi-Access Edge Computing: A Survey," *IEEE Access*, vol. 9, pp. 10903–10924, 2021, doi: 10.1109/ACCESS.2021.3050155.

[40] S. Lee, J. Ali, and B. Roh, "Performance Comparison of Software Defined Networking Simulators for Tactical Network: Mininet vs. OPNET," in *2019 International Conference on Computing, Networking and Communications (ICNC)*, Feb. 2019, pp. 197–202. doi: 10.1109/ICCNC.2019.8685572.

[41] P. Krongbaramee and Y. Somchit, "Implementation of SDN Stateful Firewall on Data Plane using Open vSwitch," in *2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Jul. 2018, pp. 1–5. doi: 10.1109/JCSSE.2018.8457354.