

QoSFlow: Prioritizing Flows in Software-Defined Networking for Enhanced Quality of Service

Dourahamane IDE BARKIRE, Benjamin KONÉ, Jean Pierre LENO, Ahmed Dooguy KORA

E-INOV LAB, Ecole Supérieure Multinationale des Télécommunications (ESMT), Dakar, Sénégal

dourahamane.barire.edu@esmt.sn, benjikone@yahoo.fr, jean.lenno.etu@esmt.sn, ahmed.kora@esmt.sn

Abstract— Software-Defined Networking (SDN) represents the next generation of network infrastructure, offering flexibility and the ability to quickly integrate services tailored to specific business applications and use cases. Motivated by the rapid rise of SDN and the increasing focus on Quality of Service (QoS) within SDN networks, this project explores how SDN can replace traditional network architectures and improve QoS management. Specifically, we investigate the role of protocols such as OpenFlow, iPerf, and Open vSwitch Database Protocol, which are pivotal in configuring and managing virtual bridges, ports, tunnels, and queues in SDN environments. The project emphasizes the QoSFlow solution for effectively managing QoS in SDN networks, demonstrating its potential for enhancing flow prioritization and overall network performance.

Keywords— IP Network; OpenFlow; QoS; QoSFlow; SDN; Streaming

I. INTRODUCTION

Current communication networks suffer from inflexibility and the inability to adapt to the requirements of transmitted network traffic. In addition, with the increase in network traffic, communication network operators need to introduce new flexible and dynamic resource management methods to provide the best possible performance for their networks. With the emergence of Software-Defined Networking (SDN), telecom operators have tools that allow them to administer and program the network more easily. In fact, SDN integrates automatic configuration, enabling network administrators to quickly configure, administer, secure, and optimize networks with dynamic and automated SDN programs [1].

This is made possible by the fact that in SDN, the control plane is separated from the data plane. Furthermore, with the growth of the Internet, new types of network applications and services are accessible. The present article focuses on Quality of Service (QoS) management in SDN networks based on SDN controllers [2],[3]. To this end, after a detailed overview of SDN networks and QoS management, we present a comparative study of QoS solutions and an experiment on QoS in SDN networks [4].

In our research and experimentation toward a solution, we have studied the effects on network traffic when applying dynamic management and QoS in OpenFlow SDN networks, as well as data transfer protocols such as Transmission Control Protocol (TCP)/User Datagram Protocol (UDP). Our contributions include an experiment designed to provide

insights into the role of dynamic QoS management in optimizing network traffic flow. The findings from this work aim to equip network administrators with the knowledge needed to make more informed decisions that enhance network performance.

The rest of this article is structured as follows. First, we give a basic technical presentation of SDN and the concept of QoS in IP networks. Then, we detail our evaluation methodology and experimental results. Finally, we summarize the contributions of this study and prospects for future work.

II. TECHNICAL BACKGROUND

A. SDN and OpenFlow

SDN is a networking concept that involves a clear separation between the data plane and control plane of the network. SDN makes the network programmable, which means that applications can interact directly with the network, and defining a network management policy involves writing programs and deploying them in controllers [5].

An omniscient central controller enables network engineers to implement unique and flexible transfer policies, with the only limitations being related to the capacity of the software operating the controller. If the controller is considered a middleware, the applications that tell the controllers how to program the network use northbound interfaces, while the application programming interfaces (APIs) and protocols used by the controller to communicate with network equipment are referred to as southbound interfaces.

Here are some advantages of using an SDN network [6]:

- **Programmability:** Network administrators typically configure network equipment using a command-line interface or the equipment's graphical user interface. By making networks programmable, SDN aims to change this by providing administrators with APIs that allow them to program network devices using multiple languages.
- **Broader Accessibility:** The use of APIs also means that network programming is no longer limited to just network engineers who are capable of using a command-line interface (CLI) but is accessible to a range of tools.
- **Integration with Orchestration Tools:** Orchestration tools could integrate network provisioning tasks with other tasks required to create a business application.

B. SDN architecture

The SDN architecture (see Figure 1) is structured in several interdependent layers, each playing a specific role in ensuring the programmability, centralised management and optimisation of networks.

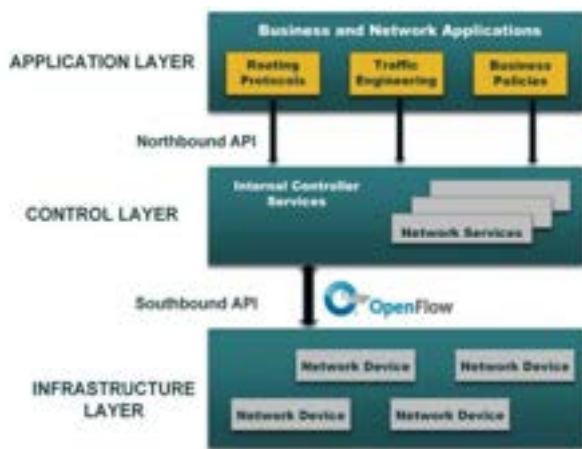


Figure 1. SDN layers

1) **The infrastructure layer** consists of physical and virtual network devices. In the case of a virtual switch, this packet processing function is performed by software. The abstraction layer embodies one or more flow tables.

2) **The control layer** is based on the SDN controller, which provides a global view of the network and infrastructure devices (the core of the SDN network). It generates network configurations based on policies defined by the network operator and receives instructions via the northbound API, and its documentation is published to allow interfacing with the southbound APIs [7]–[9].

3) **The application layer** includes one or more network applications that interact with the controllers to use an abstract view of the network for their internal decision-making processes. An SDN application installed on top of the controller includes SDN application logic and an Application–Controller Plane Interface (A-CPI) driver.

C. OpenFlow

OpenFlow is standardized by the Open Networking Foundation and is the protocol that describes the interaction of one or more control servers with OpenFlow-compatible switches [5], [10].

1) **An OpenFlow switch** is defined by one or more flow tables. Each table is equivalent to a set of flows, each of which consists of a list of discriminators related to the content of a frame, associated with processing actions to be applied to the corresponding frames [5].

2) **OVSDB protocol:** The Open vSwitch Database Protocol (OVSDB) allows programming of the Open vSwitch database (see Figure 2). This database contains the configuration of an

Open vSwitch daemon that describes the switching behavior of a virtual switch, not the behavior or configuration of a routing system. In the case where the DB-Schema is extended in a future version to cover routing system elements, implementers and operators must refer to the work of the Internet Engineering Task Force (IETF) and the Interface to the Routing System (I2RS) that specifies the protocols and data models for temporal or event-based interaction with the routing system [11], [12], [13].

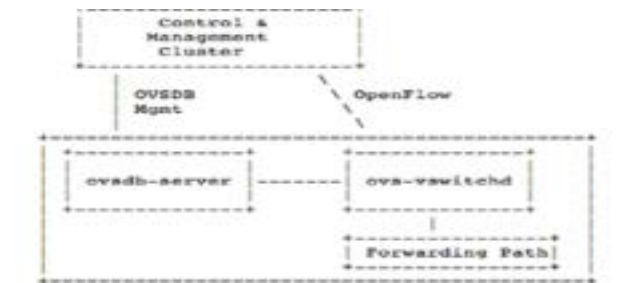


Figure 2. OVSDB protocol architecture

D. Quality of Service Concept in IP Network

With the growth of the Internet, new types of applications and network services have emerged for end users. However, all these applications require different treatments for their own flows in order to achieve successful communication in the network. Addressing these requirements requires a well-defined QoS mechanism in the network. QoS factors such as service reliability, service availability, service functionality, latency, scalability, efficiency, service quality, packet loss rate, jitter, throughput, and delay play an important role in how a particular network can handle and manage packets in a network. Since the role of SDN is to allow the network operating system to acquire greater governance of the control plane within the network, and furthermore since SDN is logically centralized, controllers have global visibility over the entire network unlike conventional networks. In addition, QoS provisioning at the flow or application level becomes easier and more manageable for network administrators [7], [14], [15].

Some QoS management models include:

- The IntServ model
- The DiffServ model
- Multi-Protocol Label Switching MPLS

1) **QoSFlow:** QoSFlow is a QoS management solution for SDN networks. It enables QoS control using a flow-based approach rather than IP addresses. Figure 3 illustrates the high-level architecture of FlowQoS. Users specify the maximum allowed bandwidth for particular high-level applications in the home network using a web configuration tool. This tool then creates the rate shaping configuration, which has two main components, namely the flow classifier and the SDN-based rate controller [8]. The flow classifier contains a lookup table where the key is a flow tuple composed of the source IP address, destination IP address, protocol, source port, and destination port. This component uses two modules to perform traffic

classification. The first module performs early identification of HTTP and HTTPS traffic. The second component, the SDN-based rate controller, performs identification of other flows.

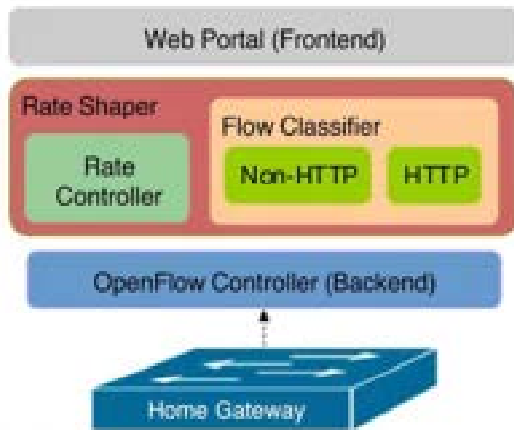


Figure 3. Quality of Service Framework (QoSF) architecture

2) A comparison of QoS management solutions in SDN networks is shown in Table 1.

Table 1 presents a comparative overview of various QoS management solutions implemented in SDN networks based on specific criteria.

TABLE 1. COMPARATIVE OVERVIEW OF QoS MANAGEMENT SOLUTIONS IN SDN NETWORKS

Solutions Criteria	FlowQoS	PolicyC op	OpenQoS	QoSFlow	ROIA
Traffic classification and identification	Flow-based	Class-based	Flow-based	Class-based	✓
QoS mechanism for differentiated treatment	Resource reservation	SLA-based QoS policy management	Optimisation of the path by flow Dynamic routing with QoS	Packet scheduling and queuing	QoE sensitive mechanism
Implementation in the SDN architecture	Implementation plan	Implementation plan	Control plan	Control plan	North bound API
QoS parameters	Bandwidth Latency		Minimises packet loss Minimises latency	Queue Bandwidth	Response time Minimises packet loss
Nature of QoS management	static	dynamic	dynamic	Dynamic	dynamic

III. EXPERIMENTATION AND ANALYSIS OF RESULTS

A. Technological Choice

In this part, we will first choose the SDN controller. Then, an exploration of the SDN controller will be carried out with its

different versions based on the version used in the project. Next, we will explain the choice of the Mininet network emulator and iPerf, the tool that allows us to examine the accuracy of measurements, and the OpenvSwitch switch.

1) **The ODL Controller** is an open-source collaborative platform that aims to accelerate the adoption of SDN. Its role is to provide centralized and programmable control, as well as open-source network device monitoring protocols. We used its Beryllium version in our work [16] [17]. The Beryllium version of the ODL controller offers better compatibility with OpenFlow, which is essential for fine-tuned traffic management in our test configuration.

2) **Mininet** is an open-source network emulator with the particularity of including SDN controllers with OpenFlow support. It runs on the Linux kernel and uses Python for its API, allowing for emulation of a complete network configuration [16] [17]. It is ideal for simulating complex networks in a controlled environment.

3) **OpenvSwitch** is an open-source software implementation of an Ethernet switch providing level 2 and 3 routing of the OSI model for virtualized environments supporting different protocols and standards. In our work, it allowed us to communicate between Linux virtual machines [11], [12]. It can simulate realistic network environments by supporting various protocols required for QoS analysis.

4) **Iperf** is a live network measurement tool whose main goal is to determine the maximum achievable throughput on the network. It can be used for other measurements such as jitter and packet loss of UDP/TCP packets with different specified transmission rates and intervals [19], [20].

5) **Wireshark** is a free packet analyzer used in the analysis and troubleshooting of computer networks. In our context, we used it for investigation on OpenFlow messages exchanged between the OpenFlow switch and the OpenDaylight controller [16], [17].

B. Architecture and Evaluation Methodology

The figure depicts a testbed where implementation and testing related to traffic shaping, packet scheduling, and enqueueing in an SDN network will be performed. We aim to create a network composed of an OpenDaylight controller, four switches, one OpenvSwitch where our QoS will be applied, and four hosts using the Mininet simulator. Each of OpenvSwitch's outgoing links holds a bandwidth of its own. Each OpenvSwitch is connected to the OpenDaylight controller via their eth0 management interface, which is dedicated solely to southbound communication with the controller. OpenDaylight is connected to a machine that monitors the network and injects QoS rules using the northbound API [18].

The proposed architecture (see Figure 4), with a centralised configuration based on OpenDaylight and a network simulated using Mininet, makes it possible to implement and test QoS prioritization strategies with precise and reproducible control of network traffic. We suppose that the integration of

prioritization rules via OpenDaylight can improve traffic management and reduce latency for critical flows.

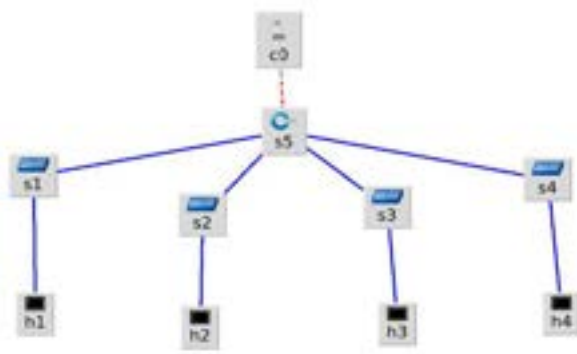


Figure 4. Experimental model

The following table (Table 2) summarizes the characteristics of the virtual machines used in this experiment, including details about their specifications and software versions.

TABLE 2. TABLE OF VIRTUAL MACHINE CHARACTERISTICS

Names	Mininet	OpenDaylight(ODL)
Sizes	64GB	64GB
Rams	4GB	4GB
OS versions	ubuntu-22.10-amd64.iso	ubuntu-22.10-amd64.iso
Mininet and ODL version	2.3.0	3.0.3
Distribution	Mininet 2.3.0/ python 3	karaf-0.4.4-Beryllium-SR4
Disk	20GB	20GB
Processors	2	2

C. Results and analysis

After confirming that our devices are functioning correctly and that connectivity is established, this section focuses on analyzing results using Mininet, iPerf3, and NETEM tools [21]. These tools help in understanding key network performance metrics such as throughput versus bandwidth, transfer protocols, Linux queuing mechanisms (qdisc), and network delay. We conducted verifications before and after introducing or distributing network delays. The analysis covers network emulation with Mininet and iPerf3, followed by a study of delay effects using Mininet and NETEM tools.

1) Emulation of our SDN network: The bandwidth is a physical property of the transmission medium that is influenced by factors such as cable or fiber construction and length, representing the maximum data capacity of a channel (measured in bits per second or bps). In contrast, throughput refers to the actual data transfer rate achieved between two endpoints over a network path. Throughput measurements are particularly insightful for diagnosing network issues like latency and packet loss.

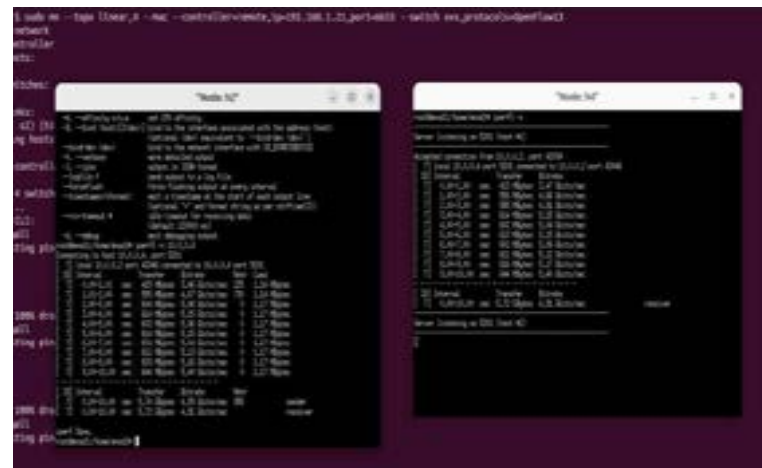


Figure 5. Launching client and server terminals

The final output indicates a total data transfer of 5.72 GB, with a useful throughput of 4.91 Gbps. The command "iperf3 -c 10.0.0.4 -t 5" configures the client to transmit packets to the server at an average rate of a few Gbps over a 5-second interval.

Initially, we utilized the default TCP protocol to measure throughput. Recognizing that various data transfer protocols exist at the transport layer of the OSI model, we transitioned from TCP to UDP. The server automatically detected this switch with iPerf3, and the results with the UDP protocol were promising, as shown in Figure 6.

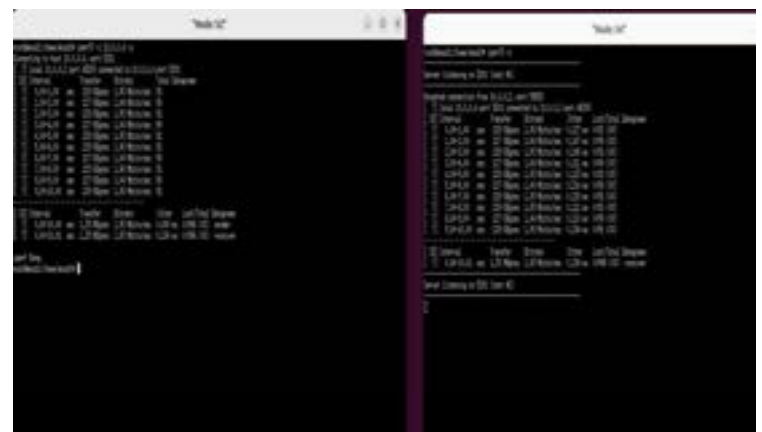


Figure 6. Specification of the transport layer protocol used by host h2 to send UDP datagrams

After the test, a summary showed a total of 1.25 MB of data transferred, with a maximum bandwidth of 1.05 Mbps over 10 seconds. Jitter—indicating delay variability between packets—was 0.134 ms, with zero datagram loss and a total of 906 received datagrams, yielding a 0% loss rate. These metrics are also visible on the server. The server port can be adjusted with the following command: iperf3 -s -p 3250 (for h4) and iperf3 -c 10.0.0.4 -p 3250 (for h2).

2) Emulation of our network with NETEM: In this section, we employ NETEM, a Linux-based emulator, to simulate real-world application performance on a virtual network. This virtual environment allows for testing and assessing network

behavior under various conditions, from the application layer to the data link layer. NETEM provides capabilities to manipulate parameters such as delay, jitter, packet loss, duplication, and packet reordering.

After identifying connected network interfaces and introducing a 100ms delay, the results shown in Figure 7.

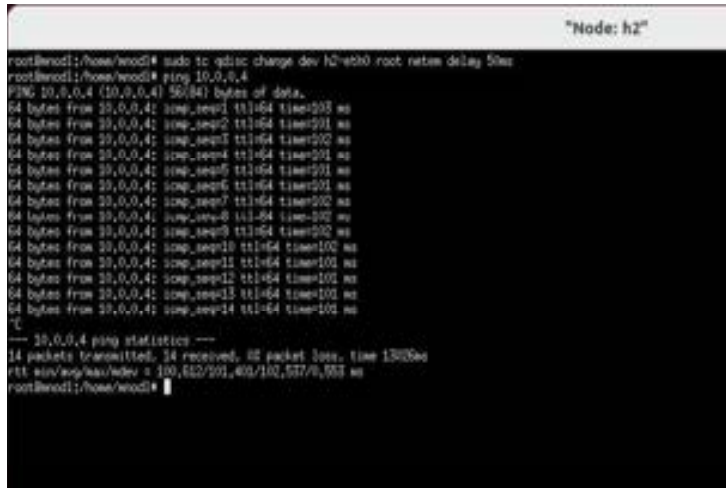
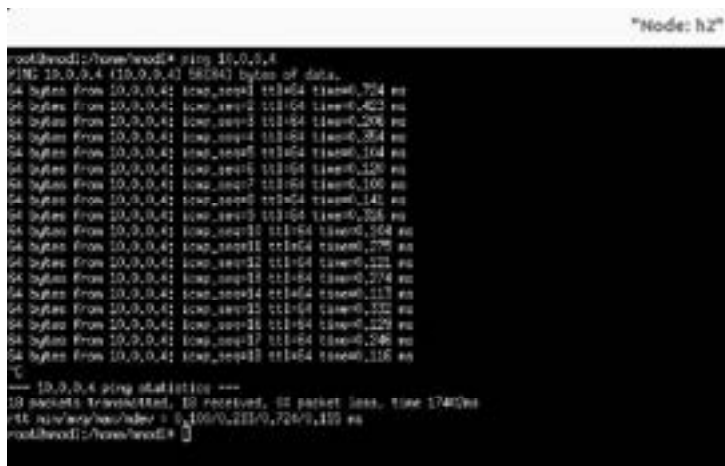
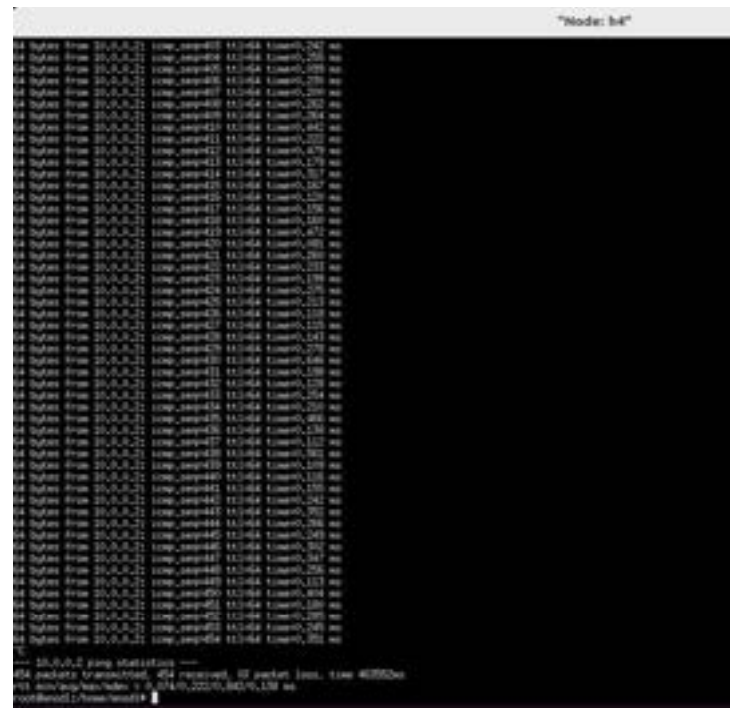


Figure 7. Verification of latency after emulating a 100ms delay using ping

The results indicate that all 14 packets were received without loss (0% packet loss), and the round-trip time (RTT) metrics were as follows: minimum 100.612 ms, average 101.401 ms, maximum 102.537 ms, and standard deviation 0.553 ms. These values can be reset using *"sudo tc qdisc del dev h2-eth0 root netem"* Once we removed the delay rules, as shown in Figure 8.



(a)



(b)

Figure 8. Verification of latency after removing all rules on both devices.

The measurements show that all 18 packets were received with no packet loss, and the RTT metrics were as follows: minimum 0.100 ms, average 0.233 ms, maximum 0.724 ms, and standard deviation 0.155 ms. When introducing a jitter parameter with a 100ms delay, packets traveling from the client to the server's connected interface experienced a 100ms delay, with a random variation of $\pm$ the jitter value.

The results shown in Figure 9 that all 24 packets were successfully received (0% packet loss), with RTT values as follows: minimum 9.623 ms, average 101.821 ms, maximum 110.796 ms, and standard deviation 6.029 ms.

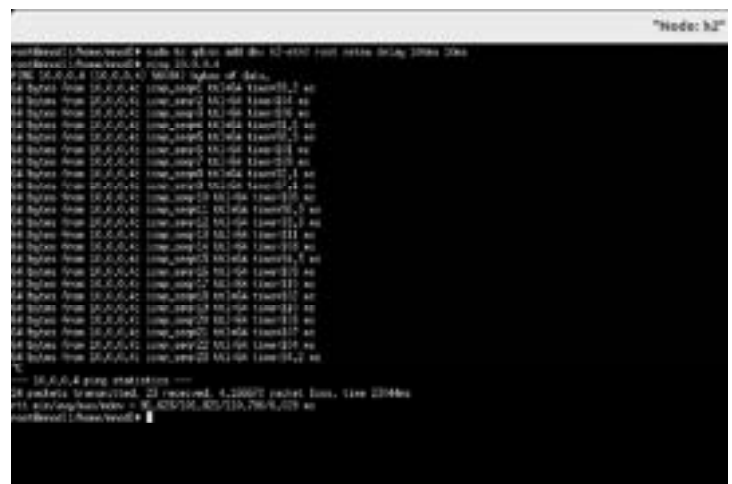


Figure 9. Verification of RTT after adding a 100-millisecond delay and a 10-millisecond jitter on the h2-eth0 interface.

NETEM enables us to specify a distribution model for delay variations, as delays in networks are often non-uniform. To simulate more realistic network conditions, we applied a normal distribution:

```
"sudo tc qdisc add dev h2-eth0 root netem" delay 100ms
30ms distribution normal
```

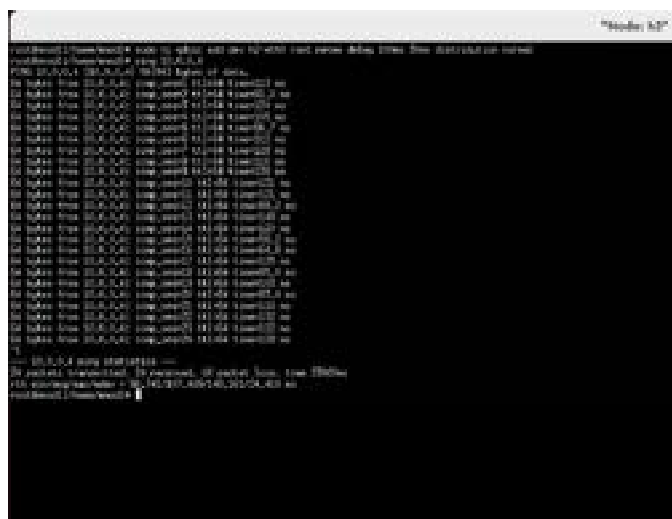


Figure 10. Latency verification using normal distribution

The results show (see Figure 10) that all 24 packets were successfully received (0% packet loss), with RTT values as follows: minimum 56.741 ms, average 107.409 ms, maximum 148.161 ms, and standard deviation 24.410 ms.

To conclude our emulation tests, we added packet duplication. Duplicate packets often result from retransmissions. With NETEM's "duplicate" option, we achieved the results shown in Figure 11. The network performance was assessed using a ping test to measure packet loss and latency.

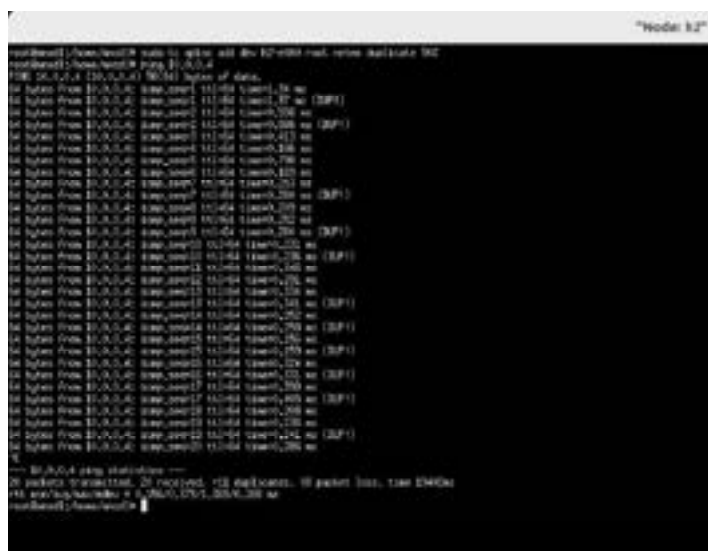


Figure 11. Ping test illustrating the effect of packet duplication

The output indicates that 11 duplicate packets were received, marked as (DUP!). The duplication rate tends to stabilize around the configured level with additional test iterations.

IV. CONCLUSIONS AND FUTURE WORK

In this preliminary study, we thoroughly reviewed the state of the art in SDN networks and the associated protocols, focusing on QoS requirements. We presented a comparative analysis of end-to-end QoS management solutions within SDN networks, demonstrating the potential for enhanced network performance through effective QoS strategies. The emulation tests that we conducted validated the feasibility and effectiveness of our proposed architecture for meeting QoS demands, particularly through tools such as Mininet, iPerf3, and NETEM.

Our findings revealed significant insights into the management of network performance, especially regarding throughput, latency, and packet loss. These results underscore the potential of using SDN to optimize network behavior dynamically and adaptively. Specifically, the ability to manage delay at the path or link level emerged as a crucial area for improvement, highlighting avenues for future exploration.

In terms of future work, we would like to further investigate the topic by implementing the studied QoS management solutions, particularly QoSFlow combined with AI or machine learning, to evaluate their performance according to relevant and key criteria for improving QoS in a network.

Overall, our study lays the groundwork for further advancements in SDN and QoS, offering practical implications for network administrators. The insights gained from our experiments can support informed decision making, guiding adjustments that enhance network reliability and efficiency. This research represents a step forward in the ongoing journey toward more resilient, adaptive, and intelligent network infrastructures.

REFERENCES

- [1] I. Choukri, M. Ouzzif, and K. Bouragba, "Software Defined Networking (SDN): Etat de L'art." [Online]. Available: <https://hal.science/hal-02298874>
- [2] K. Alghamdi and R. Braun, "Software Defined Network (SDN) and OpenFlow Protocol in 5G Network," *Commun. Netw.*, vol. 12, no. 01, pp. 28–40, 2020, doi: 10.4236/cn.2020.121002.
- [3] V. Tiwari, "SDN and Openflow for Beginners with Hands On Lab," p. 108, 2016.
- [4] 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.
- [5] "The basics of SDN and the OpenFlow Network Architecture - NoviFlow." <https://noviflow.com/the-basics-of-sdn-and-the-openflow-network-architecture/> (accessed May 07, 2023).
- [6] Open Networking Foundation, "Software-Defined Networking (SDN) Definition - Open Networking Foundation," Open Networking Foundation, 2018. <https://opennetworking.org/sdn-definition/> (accessed May 07, 2023).
- [7] "A Survey on Autonomic Provisioning and Management of QoS in SDN Networks. IEEE Access, 1–1 | 10.1109/ACCESS.2019.2919957." <https://sci-hub.se/https://ieeexplore.ieee.org/document/8726285> (accessed May 07, 2023).
- [8] O. Opencommons@uconn and H. Ghalwash, "QoS in SDN-Based Large-Scale Networks," 2020, Accessed: May 07, 2023. [Online]. Available: <https://opencommons.uconn.edu/dissertations/2482>

- [9] A. S. Parihar and N. Tiwari, "Comparative study of fault-diagnosis models based on qos metrics in SDN," Lect. Notes Data Eng. Commun. Technol., vol. 66, pp. 771–784, 2021, doi: 10.1007/978-981-16-0965-7_60/COVER.
- [10] "https://www.academia.edu/25862165/FlowQoS." <https://www.academia.edu/25862165/FlowQoS> (accessed May 07, 2023).
- [11] "ovs-vswitchd.conf.db.5 (1)".
- [12] "Open vSwitch." <http://www.openvswitch.org/> (accessed May 07, 2023).
- [13] B. Pfaff and B. Davie, "The Open vSwitch Database Management Protocol," Dec. 2013, doi: 10.17487/RFC7047.
- [14] A. Binsahq, T. R. Sheltami, and K. Salah, "A Survey on Autonomic Provisioning and Management of QoS in SDN Networks," IEEE Access, vol. 7, pp. 73384–73435, 2019, doi: 10.1109/ACCESS.2019.2919957.
- [15] "A Survey on Autonomic Provisioning and Management of QoS in SDN Networks | IEEE Journals & Magazine | IEEE Xplore." <https://ieeexplore.ieee.org/document/8726285> (accessed May 07, 2023).
- [16] "Enterprise Open Source and Linux | Ubuntu." <https://ubuntu.com/> (accessed May 07, 2023).
- [17] "Platform Overview - OpenDaylight." <https://www.opendaylight.org/about/platform-overview> (accessed May 07, 2023).
- [18] "Mininet Tool | Mininet for SDN (Support) | How to implement SDN | Network Simulation Tools." <https://networksimulationtools.com/how-to-implement-sdn/> (accessed May 07, 2023).
- [19] "iPerf - Public iPerf3 servers." <https://iperf.fr/iperf-servers.php#public-servers> (accessed May 07, 2023).
- [20] "https://iperf.fr/iperf-doc.php#3doc." <https://iperf.fr/iperf-doc.php#3doc> (accessed May 07, 2023).
- [21] "Our Labs and Groups - College of Engineering and Computing | University of South Carolina." https://www.sc.edu/study/colleges_schools/engineering_and_computing/departments/electrical_engineering/our_research/our_labs/index.php (accessed May 07, 2023).

in Quality of Service (QoS) management for Software-Defined Networks (SDN), a theme he continues to pursue with passion. Jean Pierre currently works at Sonatel (Orange Senegal) as an agent specializing in the protection of FTTH, ADSL, and fixed 5G networks. He has significant experience in deploying network services such as Internet, VoIP, and IPTV on platforms like Huawei, Nokia, and ZTE.



Ahmed Dooguy Kora is an IEEE Senior Member and a consultant for the International Telecommunication Union (ITU). He received a Master's degree in Networks and Telecommunications from the Ecole Supérieure Multinationale des Télécommunications (ESMT) in 2003 and completed his Ph.D. in Telecommunications at the University of Limoges, France, in 2007. He is currently a Professor at ESMT, where he serves as the Head of Teaching, Training, and Research. His research interests include communication and network system architecture (2G to 6G), free-space optics, fiber optics, quality of service, universal access, artificial intelligence, Software-Defined Networking (SDN), Cloud RAN, and cognitive radio.



Ide Barkire Dourahamane is a PhD student in Telecommunications at the École Supérieure Multinationale des Télécommunications (ESMT), Dakar, Senegal, and Université Cheikh Anta Diop. He earned his Master's degree in Networks and Telecommunications from ESMT in 2011 and holds a Diploma in Teleinformatics Engineering from ESMT. His research focuses on dynamic spectrum management.

He is currently a department head for Digital Culture at the University of Dosso, Niger, and has worked in various academic and telecommunications roles, including as a contractual telecom department head at SGEM Telecom. He has also completed several internships in Niger and Senegal, contributing to his expertise in telecom and IT systems.



Benjamin Koné received his Ph.D. in Mathematics and Computer Science with a Telecommunications specialization from Université Cheikh Anta Diop and Ecole Supérieure Multinationale des Télécommunications, Senegal, in 2023. His research focuses on the deployment of Software-Defined Networks (SDN) and Network Function Virtualization (NFV) to improve rural connectivity. He earned a Master's degree in Networks and Telecommunications

from ESMT Dakar, with practical experience in RF planning, optimization, and quality of service for 2G, 3G, 4G, and 5G networks.

Since 2024, he has been teaching telecommunications in private universities in Mali. He has contributed to multiple international conferences and has published research on network resource management and orchestration for virtualized networks. Currently, he is a co-founder and associate researcher at Find RD, where he focuses on innovation and rural connectivity solutions.



Jean Pierre LENO is a Telecommunications Design Engineer who graduated from the École Supérieure Multinationale des Télécommunications (ESMT) in 2023. Passionate about new technologies and information security, he has developed a solid expertise