International Journal on Advanced Science Engineering Information Technology

Vol.13 (2023) No. 6
ISSN: 2088-5334

Dynamic QoS: Automatically Modifying QoS Queue's Maximum Bandwidth Rate-Limit of Network Devices for Network Improvement

Muhammad Fendi Osman [a], Mohd Rizal Mohd Isa [a], Mohammad Adib Khairuddin [a], Mohd 'Afizi Mohd Shukran [a], Noor Afiza Mat Razali [a], Nur Diyana Kamarudin [a], Amin Suharjono [b]

[a] *Computer Science Department, National Defense University of Malaysia, Malaysia*
[b] *Department of Electrical Engineering, Politeknik Negeri Semarang, Prof. Sudarto Street, Semarang, 50275, Indonesia*
*Corresponding author: *rizal@upnm.edu.my*

***Abstract*—The heterogeneous data traffic of today's network is a huge challenge to existing best-effort network technology, particularly in the context of large Ethernet, which handles hundreds to thousands of users. The existing conventional best-effort network technology is no longer efficient to handle the diversity of traffic types in the network and requires network management equipment such as Quality of Service (QOS). Usually, QOS is implemented on the gateway router. However, for better network performance and management, to guarantee high priority for sensitive traffic like video conferencing, Voice over Internet Protocol (VoIP), and streaming media within an internal network, it is nice to have QoS implemented on each router in the LAN network, starting from the access router to the gateway router. This paper is to demonstrate the effectiveness of the proposed dynamic QoS that has been developed and deployed in the LAN, purposely to provide adequate bandwidth for sensitive traffic when the network utilization is high and congested, by automatically modifying the QoS Queue's Maximum Bandwidth Rate-Limit of the best-effort traffic queue of the related router. The performance of the proposed developed dynamic QoS was evaluated via a comparison study before and after the dynamic QoS was presented in the network simulation environment that was built using Mininet. Results from the testing show that the developed dynamic QoS can improve the network's performance by automatically giving the appropriate bandwidth for sensitive traffic on the fly while needed/on demand.**

***Keywords*— Dynamic QoS; QoS; quality of service; queue; rate-limit.**

Manuscript received 18 Aug. 2023; revised 15 Sep. 2023; accepted 30 Oct. 2023. Date of publication 31 Dec. 2023.
IJASEIT is licensed under a Creative Commons Attribution-Share Alike 4.0 International License.

I. INTRODUCTION

Nowadays, real-time application users, in particular, will experience sluggishness and poor performance due to the high network utilization and diversity of traffic types in the network [1]–[3]. The existing best-effort network technology or Ethernet is no longer efficient and requires a better network management approach, such as *Quality of Service* (QoS), to provide good treatment on top of the best-effort technology.

Typically, QoS configuration is static, and priorities have to be assigned and configured before and during the implementation [4]–[8]. A waste of resources may occur if network resources are assigned with a static configuration during pre-configuration, such as the pre-allocation of a large amount of bandwidth, for instance, that is never used because there is not that type of traffic at that time, which is a waste [4].

Therefore, the need for Dynamic QoS is very practical at this point. With the presence of *Software-Defined Networking* (SDN) and its beauty of the global view and central control features, dynamic QoS can be developed/implemented, and multiple dynamic QoS in a network also can be managed and controlled from a central location [4]. Currently, many researchers doing dynamic QoS research studies aim to improve the performance of the existing networks (with different ways and different approaches).

There are many areas for QoS improvement to be explored, such as *Admission Control*, *QoS Routing*, *Resource Reservation*, *Queuing & Scheduling*, *Traffic-Shaping*, *Packet Marking & Policy*, and *Traffic Classification* [9]. This paper contributes to improvement in the QoS *Queuing & scheduling* area by proposing a Dynamic QoS that can modify the QoS Queue's Maximum Bandwidth Rate-Limit to provide adequate bandwidth for sensitive traffic. This research's

proposed dynamic QoS framework will be discussed further in the next section.

The presence of SDN allows researchers to do advanced research and allows network operators to improve the network and overcome existing network constraints [10]. SDN can be considered as a revolution in the existing networks [11]–[15]. SDN is dynamic, manageable, cost-effective, and adaptable, making it ideal for today's applications. In the SDN architecture, the control and data planes are decoupled. Network control is directly programmable, and the controller communicates with data planes via the *OpenFlow* (OF) protocol [10], [11], [16]–[20]. Once programable, it can be instructed to automate its process, such as permitting or denying traffic, monitoring, diverting traffic or rerouting, resource reservation, QOS/QoE, security, etc. This section focuses on reviewing the previous research on dynamic QoS.

The authors of [4] proposed Dynamic QoS support for IoT backhaul networks through SDN. The proposed framework uses the default QoS profile classification. The default QoS profile classification is the standard information of transmission delay, bandwidth, and packet loss of 4 types of IoT devices: *sensor*, *actuator*, *video*, and *audio*. Once triggered or meets the threshold, the proposed framework will re-configure the network for IoT devices to meet its QoS requirement. The demonstration shows that the QoS parameter is significantly improved. However, the default QoS profile classification requires frequent updates. As technology rapidly changes, the QoS parameters value of sensors, actuators, videos, and audio types also change. If not updated, the classification method will no longer be accurate to classify future traffic.

The authors of [21] proposed a QoS solution by classifying incoming network traffic using a *Differentiated Services Code Point* (DSCP) and diverting the subsequent traffic to a less congested QoS queue (*assured forwarding queue*) to prevent increasing delays of the existing QoS queue (*expedited forwarding queue*) in SDN network. The demonstration shows that the proposed solution can reduce delays and improve network utilization efficiency by fully utilizing all bandwidth and QoS queues provided to flow network traffic. However, once mixed traffic is mixed between queues, it will be difficult to measure a specific type of traffic in the network, especially the type of traffic in the assured forwarding queue that is already mixed.

In a multi-path or mesh network topology, a study by [22] proposed optimizing a network using a path-planning dynamic QoS using a machine learning (ML) model for incoming traffic classification. *The k-nearest neighbors (kNN), support vector machine (SVM), and feed-forward back-propagation neural network (FFBP) algorithm have been tested and are* working accurately to predict the right path to flow the traffic. The authors of [23] study focus on reducing packet losses and delays in mesh and on Software-Defined-IoT by optimizing *Proactive Flow Creation* (PFC) and combination with *Reactive and Proactive Flow Creation* (RPFC) to optimum path-routing. The solution proposed is without ML or deep learning (DL).

The latest study of QoS is SDN-based dynamic QoS development with ML/DL approach. In a survey on ML and DL-based QoS-aware protocols for SDN, the authors of [9] presented their findings. The study demonstrates that DL-based methodologies will unquestionably be the essential prerequisite for the future. Together, DL and SDN have the potential to transform the networking sector. DL and ML can be used extensively for tasks like traffic identification, classification, routing, QoS queuing & scheduling, traffic policy, and other QoS criteria.

[24]–[28] focused on ML for QoS traffic classification. [29]–[34] focused on DL for QoS traffic classification. [35]–[37] focused on ML/DL for network security.

Motivated by what has been discussed in this section, this paper proposed a Dynamic QoS framework designed and developed to provide adequate bandwidth for sensitive traffic by automatically modifying the QoS Queue's Maximum Bandwidth Rate-Limit on demand. The proposed framework is novel and will be developed based on the SDN and DL approach. However, DL is not presented for this paper because it is currently under development and not ready to be present. Therefore, the dynamic QoS with the DL approach will be discussed in detail further in future publications.

II. Materials and Method

The conceptual framework of the proposed Dynamic QoS of the research is shown in Fig 1.

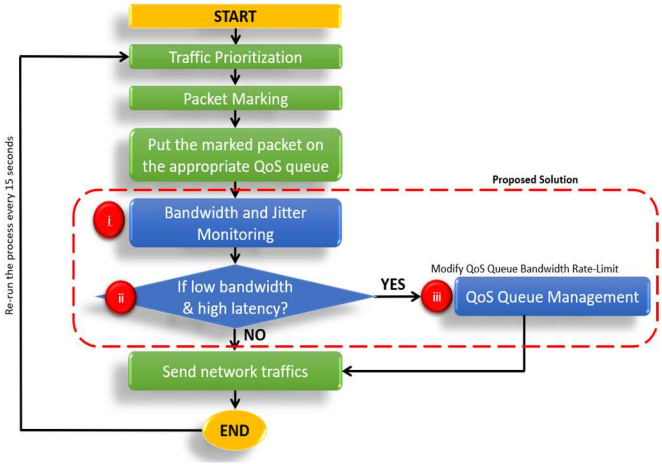

Fig. 1 Conceptual Framework

The common QoS mechanism is based on traffic classification/prioritization, packet marking, queuing, and prioritized forwarding of packets to their destinations based on priority [38]. To enhance the existing QoS and to make it dynamic, three supplementary processes are proposed: (i) *Bandwidth and Jitter monitoring*, (ii) *Analyzing Bandwidth and Latency*, and (iii) *QoS Queue Management*, as shown in Fig. 1.

The Bandwidth and Jitter monitoring processes involve the acquisition of information from network devices, particularly routers. The term ***bandwidth*** here refers to the QoS Queue's Maximum Bandwidth Rate-Limit or the maximum capacity of the specific QoS queue. Here, it captures the information of the current configured QoS Queue's Maximum Bandwidth Rate-Limit of the *best-effort traffic queue*, and captures the jitter value of the QoS queue of the *sensitive traffic queue*. ***Jitter*** represents the variation in packet delay of sensitive traffic. The higher the jitter value it presents, means the longer the delay will be.

Then, the Analyzing Bandwidth and Latency process examines the captured information from the previous process. Originally, this process utilized deep learning (DL) techniques. However, for this paper, DL is not presented. As mentioned in the previous section, this paper only focuses on clarifying the effectiveness of modifying the QoS Queue's Maximum Bandwidth Rate-Limit technique. The purpose is to test whether this technique can improve the existing network by automatically providing adequate bandwidth for sensitive traffic on the fly/on-demand or not. In the meantime, for this paper, a common IF/ELSE statement is used and examined with the threshold given usually jitter must not be more than 30 milliseconds (ms) to gain good video and voice quality [39].

If there is a demand for sensitive traffic for high bandwidth, the following process phase, QoS Queue Management, will automatically modify the QoS Queue's Maximum Bandwidth Rate-Limit of the related router. To provide adequate bandwidth for sensitive traffic, the proposed framework will reduce the QoS Queue's Maximum Bandwidth Rate-Limit of the best-effort queue and expand the QoS Queue's Maximum Bandwidth Rate-Limit of the sensitive traffic queue for every time the jitter of the sensitive traffic queue exceeds 30 milliseconds (ms). The proposed Dynamic QoS framework has been programmed to observe the network every 15 seconds (sec). The configuration of the routers will return to its default configuration if there is no/less sensitive traffic in the network path.

This proposed Dynamic QoS framework's high-level concept/approach is illustrated in Fig. 2. Condition (A) represents Default Configuration when there is no/less sensitive traffic in the network. Condition (B) represents On Demand Auto Config where the QoS Queue's Maximum Bandwidth Rate-Limit of the best-effort queue will automatically reduce and/or the QoS Queue's Maximum Bandwidth Rate-Limit of the sensitive traffic queue will automatically expand when there is a demand for high bandwidth for sensitive traffic in the network.

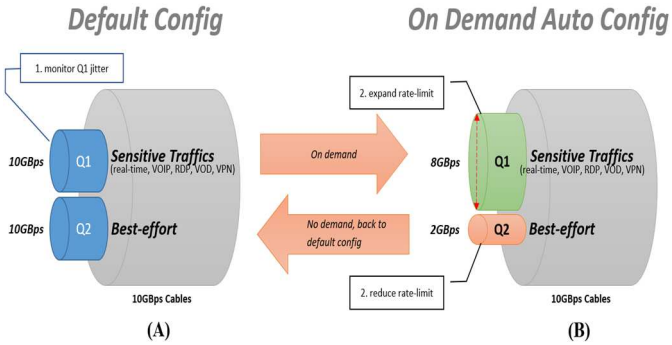

Fig. 2 Framework Approach

The proposed process flow diagram, including devices and dependencies of this research, and where the process of (i) *Bandwidth and Jitter monitoring*, (ii) *Analyzing Bandwidth and Latency*, and (iii) *QoS Queue Management* happen is shown in Fig. 3.

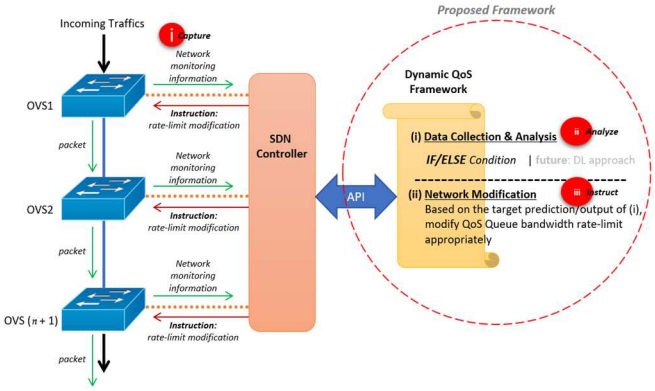

Fig. 3 Proposed Process Flow Diagram (devices & dependencies)

The methodology used for this research is as follows: -

Fig. 4 Research Methodology

In this experiment presented for this paper, for easy demonstration, only two QoS queues will be provided for each router: (i) *Q1* represents the sensitive traffic queue and (ii) *Q2* represents the best-effort traffic queue, as illustrated in Fig. 2. For the default configuration or at the initial state, both queue's maximum rate-limit has been configured as 10Gbps as the maximum size of the cable/transmission medium. With this configuration, the network works as the best-effort model, in which the network does not provide any resource guarantee or priority to the data delivery for either queue. With the proposed Dynamic QoS framework enabled, when there is a demand for sensitive traffic for high bandwidth, the proposed Dynamic QoS will provide adequate bandwidth for the Q1 traffic by automatically modifying the Q2 bandwidth rate limit of the related router.

A. Data Collection and Analysis

Data collection is the phase of collecting information from the routers. The information required from each router is (i) the *current Q2 maximum bandwidth rate-limit* information and (ii) the *Q1 jitter* value. The proposed Dynamic QoS framework then uses and analyzes the collected information to provide *preferred_config*. Prefered_config is a parameter used by the proposed Dynamic QoS framework to automatically modify the Q2 maximum bandwidth rate-limit of the related router.

A sample of pseudo-code to capture the current Q2 maximum bandwidth rate-limit for each router is shown in Fig. 5, and a sample of pseudo-code to capture the current jitter of the Q1 queue is shown in Fig.6.

```
# PREREQUISITE: The network router must be SDN enabled.
# Define the router/switch to send a GET request to.
router = http://[SDN_Controller_ID]/qos/queue/[Router_ID]

# Send a GET request and store the response.
response = requests.get(router)

# Check if the response status OK
if response = OK:
  # Print a success message.
  print("Command executed successfully.")
  print("Response:", response.text)

  # Parse the response JSON data.
  data = json.loads(response.text)

  # Extract the 'max-rate' value from the JSON data.
  max_rate = data[0]['command_result']['details']['router_port_ID']']
           ['[Q2_queue']']['config']['max-rate']

  # NOTE: 'max_rate' captured is the current Q2 maximum bandwidth
    rate-limit.

else:
  # Print an error message along with the response status code and
    text.
  print("Error message:", response.text)
```

Fig.5 Pseudo-code to capture the current *Q2 maximum bandwidth rate-limit*

```
# Pseudo-code for Bash script
while true:
  host = [router_IP_Address]

  # Run the iPerf command and store the output in a variable
  iperf_output = iperf -c [host] -p [port_number] -u -b
                 [bandwidth_size] -t [time_in_seconds]

  # Extract the jitter value from the output
  Q1_jitter = extract_latency(iperf_output)

  # Extract the throughput values from the output
  Q1_throughput = extract_ throughput (iperf_output)

  # NOTE: Capture 'throughput' is just for monitoring purposes

  # Print the extracted latency and bandwidth values
  print("Q1_Jitter:", latency)
  print("Bandwidth:", bandwidth)

  # Wait for 15 seconds before the next iteration
  # Repeat the process every 15 seconds
  sleep(15)
```

Fig. 6 To capture the current jitter of the Q1 of the router

Both output 'max_rate' and 'Q1_jitter' from Fig.5 and Fig.6 will be analyzed and examined with the threshold of 30 ms using the IF/ELSE statement (as DL is not presented in this paper). A sample of pseudo-code for this analysis is shown in Fig. 7.

```
# Initial Q2 maximum bandwidth rate-limit value in MBps
max_rate = 10000

# Initial Q1_jitter value in milliseconds
Q1_jitter = 0

# Initial preferred_config value
preferred_config = 1

# Define threshold 30 milliseconds
threshold = 30

if max_rate == 10000 and Q1_jitter > threshold:

    # Modify the Q2 maximum bandwidth rate-limit of the
    # router using OpenFlow/SDN approach for 5GBps
    # Both Q1 and Q2 will gain 5GBps guaranteed
    max_rate = 5000
    preferred_config = 2

elif bandwidthQ2 == 5000 and Q1_jitter > threshold:

    # Modify the Q2 maximum bandwidth rate-limit of the
    # router using OpenFlow/SDN approach for 2GBps
    # Q1 queue will gain 8GBps guaranteed
    # Q2 queue bandwidth will be limited up to 2GBps
    max_rate = 20000
    preferred_config = 3

elif bandwidthQ2 == 20000 and Q1_jitter > threshold:

    # No modification made, it should be maintained at GBps
    pass

else:
    # Return to default configuration
    max_rate = 10000
    preferred_config = 1

print("Modified max_rate:", max_rate)
print("Modified preferred_config:", preferred_config)
```

Fig. 7 Analyze 'max_rate' and 'Q1_jitter'

The proposed Dynamic QoS then used results from Fig. 7 to modify the Q2 maximum bandwidth rate-limit of the related router.

B. Experimental Design

The Experimental Design phase involved multiple numbers of processes as follows: -

- Design the proposed Network Optimization Framework (a dynamic QoS framework) algorithm.
- Design the simulation environment that is suitable to run and test the proposed Dynamic QoS framework.
  - The proposed network topology is shown in Fig. 8.
- Determine the evaluation criteria and parameters.
  - *Q2 maximum bandwidth rate-limit* and Q1 *Jitter* for the proposed framework input parameter.
  - *Prefered_config* is the output parameter produced for the proposed Dynamic QoS framework, automatically modifying the QoS queue's maximum bandwidth rate-limit of the related router.
- Evaluate the measurement framework.
  - The performance of the proposed framework will be evaluated via a comparison study (before and after the

proposed Dynamic QoS framework is presented). Before the proposed framework is presented, the simulation environment will represent a conventional network. Comparison of *throughput* (actual rate of successful data transfer over the QoS queues) and jitter will be measured for every millisecond (ms) up to 120 ms. The results will be tabulated and graphed for easy visualization of the differences.

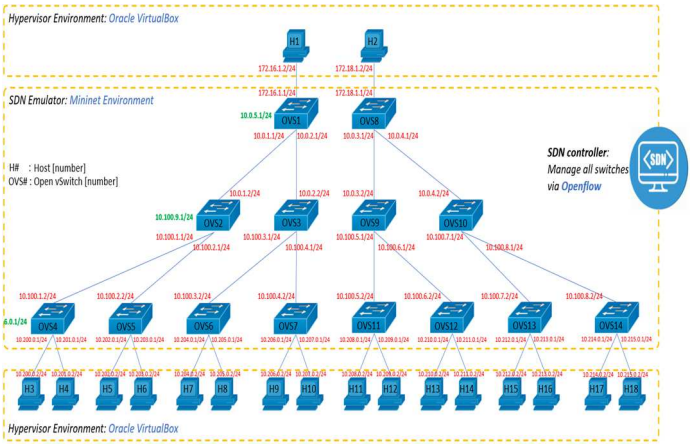

Fig. 8 Proposed Network Topology

C. Implementation

The Implementation phase involved two processes: -

- Development of the proposed Dynamic QoS framework based on the framework that was designed from the Experimental Design phase.
  - The source code of the proposed framework is written in Python with the combination of the script presented in Fig.5, Fig.6, and Fig.7.
- Development of a simulation environment based on what was proposed in the Experimental Design phase.
  - The demonstration of the proposed Dynamic QoS framework is done in a simulation environment. The simulation environment was built using MiniEdit (Graphical Mininet) which will represent as an Enterprise Network Infrastructure as shown in Fig.9. For this demonstration, only three routers are involved and each of them represents (i) the access router, (ii) the distribution/core router, and the gateway router as shown in Fig.10.

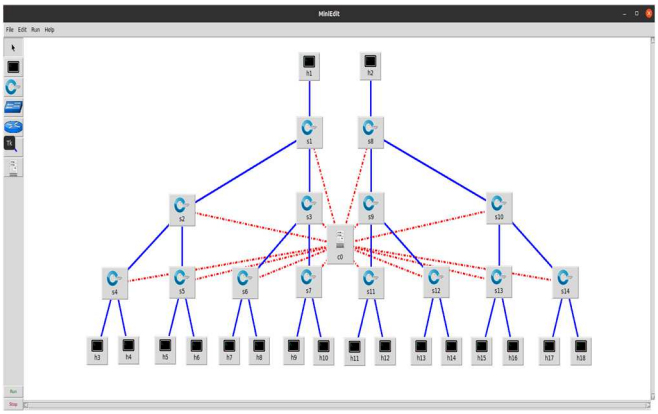

Fig. 9 Deployed Network Topology on MiniEdit (Graphical Mininet)

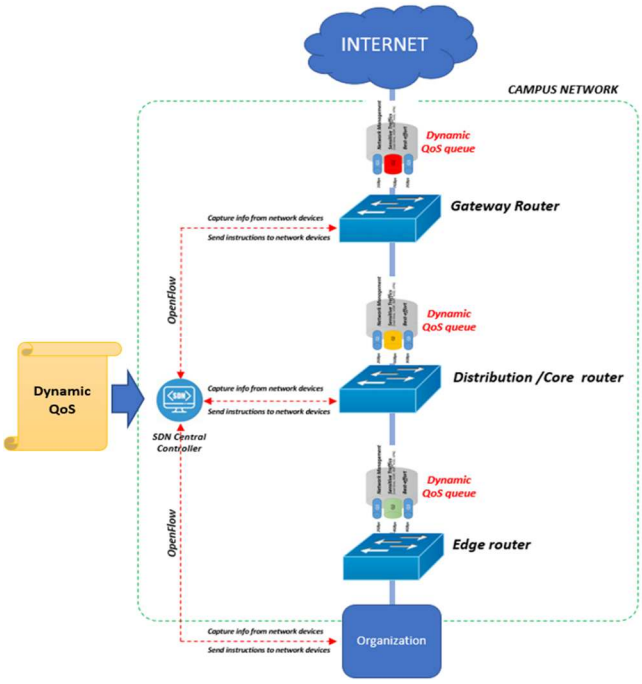

Fig. 10 Simulation/Testing Environment

D. Testing

The performance of the proposed Dynamic QoS framework will be evaluated via a comparison study (before and after the framework is presented). Two tests will be conducted: -

TEST 1: Throughput and Jitter evaluation.

Step 1: Enable the Proposed Dynamic QoS framework program.

Step 2: Run two iPerf packet generators simultaneously. One of them will represent sensitive traffic (run it for 100 ms) and the other will represent best-effort traffic (run it for 120 ms). The proposed framework uses differentiated service as a classification technique. It classifies based on a 6-bit Differentiated Services Code Point (DSCP) that is located in the IP header.

Step 3: Capture the value of *Q2 maximum bandwidth rate-limit* and *Q1 jitter* for every millisecond and tabulate it into a graph.

Step 4: Repeat Steps 2 to 3 without enabling the Proposed Dynamic QoS Framework program.

Step 5: Compare the observed differences.

TEST 2: Speed test evaluation.

Step 1: Enable the Proposed Dynamic QoS framework program.

Step 2: Send a file that simulates sensitive traffic by travel from source to destination using the sensitive traffic queue for each router.

Step 3: Capture the speed value measured in bits per second (bps)/ kilobits per second (Kbps) or Megabits per second (Mbps) and tabulate it into a graph.

Step 4: Repeat Step 2 to Step 3 by sending other files with varying sizes and bandwidth requirements.

Step 5: Repeat Steps 2 to 4 without enabling the Proposed Dynamic QoS Framework program.

Step 6: Compare the observed differences.

III. Results and Discussion

The proposed Dynamic QoS Framework has been programmed to monitor and modify the network every 15 seconds. Fig.11 and Fig.12 illustrate Test 1 enabled by the proposed Dynamic QoS framework.

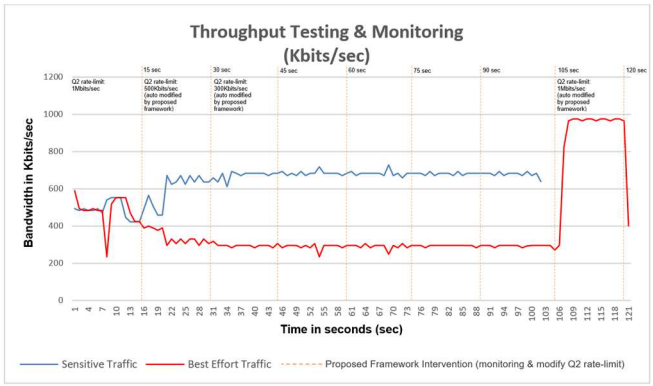

Fig. 11 Test 1(1) with Network Optimization Framework Enabled

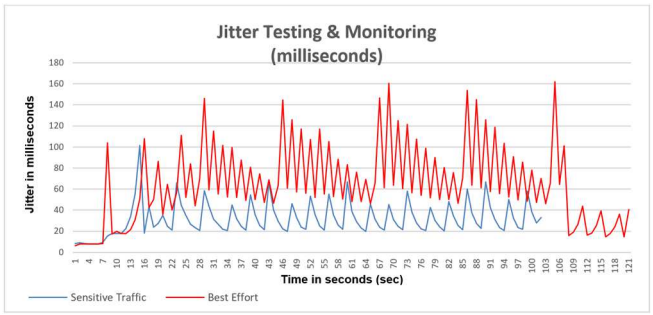

Fig. 12 Test 1(2) with Network Optimization Framework Enabled

As shown in the graph, at the 15th-second mark, when the proposed framework found that the average jitter of sensitive traffic (Q1) exceeds 30 ms, the framework automatically modified and reduced the QoS queue's maximum bandwidth rate-limit of best-effort traffic (Q2) from 1Mbits/sec to 500Kbits/sec. The graph clearly shows that the sensitive traffic gained a greater throughput after the modification, while the best-effort traffic experienced a significant drop in throughput. ***Throughput*** represents the actual rate of successful data transfer over the QoS queues. Once again, when the framework found that the average jitter of Q1 exceeds 30 ms at the 30th-second mark, the framework automatically modified the QoS queue maximum bandwidth rate limit of Q2 from 500Kbits/sec to 300Kbits/sec. The graph shows that the throughput of sensitive traffic slightly increases, remains constant, and stable until it reaches completion at the 100th-second mark. At the 105th-second mark, there is no sensitive traffic. The jitter of Q1 is 0, and the framework knows that there is no sensitive traffic at that time and responds immediately by modifying the QoS queue's maximum bandwidth rate-limit of Q2 from 300Kbits/sec to 1Mbits/sec (default configuration). The graph indicates that best-effort traffic experiences an increase in throughput after the 105th-second mark.

Compared to Fig.13 and Fig.14, no special treatment is given to the bandwidth for both traffic. Both traffics have been entertained equally. Less impact for the best-effort traffic user because it can tolerate delay, but for the sensitive traffic, the user will face slowness and poor performance; as mentioned by Cisco press, the average jitter should be less than 30 ms to gain good video and voice quality [39].

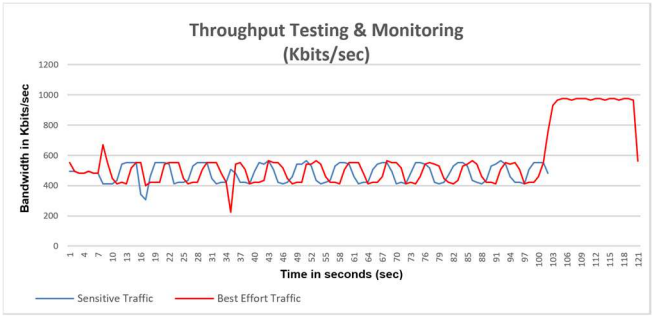

Fig. 13 Test 1(1) without Network Optimization Framework Enabled

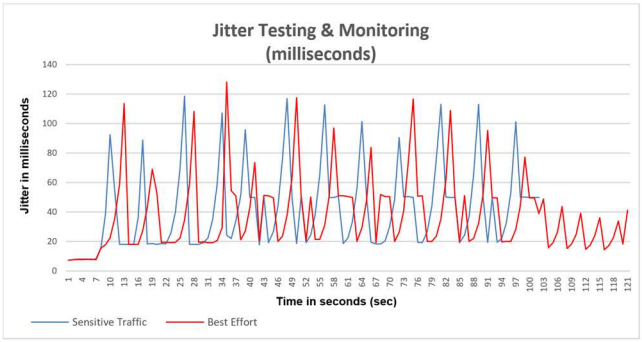

Fig. 14 Test 1(2) without Network Optimization Framework Enabled

The differences throughput gained of sensitive traffic with/without Dynamic QoS framework are shown in Fig. 15. It is about 20% higher with framework enabled. The differences jitter captured of sensitive traffic with/without Dynamic QoS framework are shown in Fig. 16. It shows that jitter of sensitive traffic with framework enabled is lower than without framework enabled. The average jitter of the sensitive traffic with framework enabled from 15th-second mark to 100th-second mark is 28.8 ms. It also means that the sensitive traffic travels to it destination with low delay and acceptable as it travels at jitter lower than 30 ms.

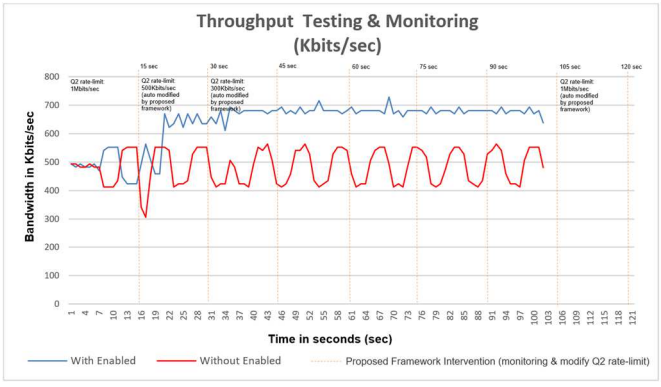

Fig. 15 Throughput gained with/without framework enabled

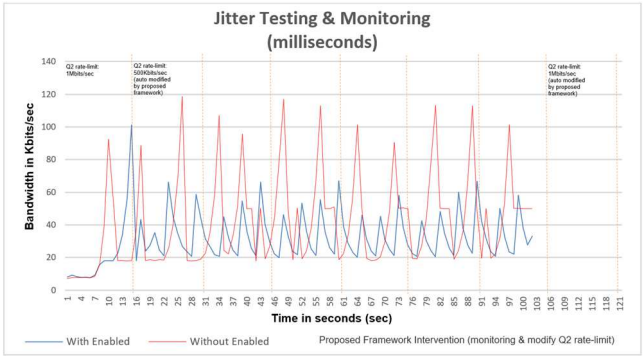

Fig. 16 Captured jitter of with/without framework enabled.

From Test 1, it can be concluded and summarized that the proposed Dynamic QoS framework can improve the performance of the network by giving the appropriate bandwidth for sensitive traffic while needed and on the fly. Test 2 is currently in progress. The findings and outcomes of Test 2 will be discussed further in a future publication.

IV. Conclusion

In summary, this paper aimed to design and develop a Novel Network Optimization Framework (a dynamic QoS program) based on the SDN and DL approach that will automatically provide appropriate bandwidth for the sensitive traffic QoS queue for each router in the network. The framework automatically adjusts the bandwidth allocation by monitoring the queue's jitter. If the value of the jitter increases, it indicates there is much sensitive traffic in the buffer waiting for the queue and may cause potential delay. To overcome this issue, if the average jitter exceeds 30ms, the framework will provide more bandwidth for the sensitive traffic queue by automatically reducing the QoS queue's maximum bandwidth rate-limit of the best-effort queue. The routers' configuration will also return to its default configuration automatically if there is no/less sensitive traffic in the network. In the testing that has been conducted, it was found that the proposed framework can improve the network performance dynamically by providing adequate bandwidth for the sensitive traffic queue. It has been determined through testing that the proposed framework effectively enhances network performance by dynamically allocating sufficient bandwidth to the sensitive traffic queue.

For future work research, the deep learning model will analyze the bandwidth and latency process of the proposed framework. The Deep Learning model is under development and will be ready soon. Once ready, the Deep Learning model and its implementation on the proposed framework will be discussed further in a future publication.

References

[1] N. Z. M. Safar, N. Abdullah, H. Kamaludin, S. Abd Ishak, and M. R. M. Isa, "Characterising and detection of botnet in P2P network for UDP protocol," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 18, no. 3, pp. 1584–1595, 2020.

[2] M. F. Mustafa *et al.*, "Student Perception Study on Smart Campus: A Case Study on Higher Education Institution," *Malaysian Journal of Computer Science*, pp. 1–20, 2021.

[3] M. R. M. Isa, M. A. Khairuddin, M. A. B. M. Sulaiman, M. N. Ismail, M. A. M. Shukran, and A. A. B. Sajak, "SIEM Network Behaviour Monitoring Framework using Deep Learning Approach for Campus Network Infrastructure," *International Journal of Electrical and Computer Engineering Systems*, pp. 9–21, 2021.

[4] S. Peros, H. Janjua, S. Akkermans, W. Joosen, and D. Hughes, "Dynamic QoS support for IoT backhaul networks through SDN," in *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, 2018, pp. 187–192.

[5] Fortinet, "What is Quality of Service (QoS) in Networking?" 2022.

[6] R. Gandotra and L. Perigo, "SDVoIP—A software-defined VoIP framework for SIP and dynamic QoS," *Comput J*, vol. 64, no. 2, pp. 254–263, 2021.

[7] A. N. Abosaif and H. S. Hamza, "Quality of service-aware service selection algorithms for the internet of things environment: A review paper," *Array*, vol. 8, p. 100041, 2020.

[8] S. Messaoudi, A. Ksentini, and C. Bonnet, "SDN Framework for QoS provisioning and latency guarantee in 5G and beyond," in *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, 2023, pp. 587–592.

[9] D. Sarma and H. Kumar, "A Survey on Machine Learning and Deep Learning based Quality of Service aware Protocols for Software Defined Networks," 2021.

[10] O. N. Foundation, "Software-defined networking: the new norm for networks," *ONF White Paper*, vol. 2, pp. 2–6, 2012.

[11] J. C. C. Chica, J. C. Imbachi, and J. F. B. Vega, "Security in SDN: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 159, p. 102595, 2020.

[12] S. S. A. Gilani, A. Qayyum, R. N. Bin Rais, and M. Bano, "SDNMesh: An SDN based routing architecture for wireless mesh networks," *IEEE Access*, vol. 8, pp. 136769–136781, 2020.

[13] N. T. Hoang, H.-N. Nguyen, H.-A. Tran, and S. Souihi, "A novel adaptive east–West Interface for a heterogeneous and distributed SDN network," *Electronics (Basel)*, vol. 11, no. 7, p. 975, 2022.

[14] M. W. Nadeem, H. G. Goh, V. Ponnusamy, and Y. Aun, "DDoS Detection in SDN using Machine Learning Techniques.," *Computers, Materials & Continua*, vol. 71, no. 1, 2022.

[15] N. Ahmed *et al.*, "Network threat detection using machine/deep learning in sdn-based platforms: a comprehensive analysis of state-of-the-art solutions, discussion, challenges, and future research direction," *Sensors*, vol. 22, no. 20, p. 7896, 2022.

[16] S. Ahmad and A. H. Mir, "Scalability, consistency, reliability and security in SDN controllers: a survey of diverse SDN controllers," *Journal of Network and Systems Management*, vol. 29, pp. 1–59, 2021.

[17] Z. Fan, J. Yao, X. Yang, Z. Wang, and X. Wan, "A multi-controller placement strategy based on delay and reliability optimization in SDN," in *2019 28th wireless and optical communications conference (WOCC)*, 2019, pp. 1–5.

[18] L. Peterson, C. Cascone, and B. Davie, *Software-defined networks: a systems approach*. Systems Approach, LLC, 2021.

[19] B. P. R. Killi and S. V. Rao, "Controller placement in software defined networks: A comprehensive survey," *Computer Networks*, vol. 163, p. 106883, 2019.

[20] R. Wazirali, R. Ahmad, and S. Alhiyari, "SDN-openflow topology discovery: An overview of performance issues," *Applied Sciences*, vol. 11, no. 15, p. 6999, 2021.

[21] A. Khater and M. R. Hashemi, "Dynamic Flow Management Based on DiffServ in SDN Networks," in *Electrical Engineering (ICEE), Iranian Conference on*, 2018, pp. 1505–1510.

[22] M. Rodriguez, R. F. Moyano, N. Pérez, D. Riofrio, and D. Benitez, "Path Planning Optimization in SDN Using Machine Learning Techniques," in *2021 IEEE Fifth Ecuador Technical Chapters Meeting (ETCM)*, 2021, pp. 1–6.

[23] N. N. Josbert, H. N. Joyce, J. Wang, and M. J. Bosco, "End-to-end QoS Routing Scheme in Industrial Internet of Things Managed by Software-Defined Networking Platform," in *2021 IEEE International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI)*, 2021, pp. 542–549.

[24] A. I. Owusu and A. Nayak, "An intelligent traffic classification in sdn-iot: A machine learning approach," in *2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2020, pp. 1–6.

[25] V. Deart, V. Mankov, and I. Krasnova, "Development of a Feature Matrix for Classifying Network Traffic in SDN in Real-Time Based on Machine Learning Algorithms," in *2020 International Scientific and Technical Conference Modern Computer Network Technologies (MoNeTeC)*, 2020, pp. 1–9.

[26] N. P. K. Goud, G. S. C. Reddy, and A. Maryposonia, "Traffic Classification of SDN Network using Machine Learning Algorithms,"

in *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2022, pp. 1181–1185.

[27] M. Shafiq, X. Yu, A. A. Laghari, L. Yao, N. K. Karn, and F. Abdessamia, "Network traffic classification techniques and comparative analysis using machine learning algorithms," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, 2016, pp. 2451–2455.

[28] M. Nsaif, G. Kovásznai, M. Abboosh, A. Malik, and R. de Fréin, "ML-Based Online Traffic Classification for SDNs," in *2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS)*, 2022, pp. 217–222.

[29] A. Malik, R. de Fréin, M. Al-Zeyadi, and J. Andreu-Perez, "Intelligent SDN traffic classification using deep learning: Deep-SDN," in *2020 2nd International Conference on Computer Communication and the Internet (ICCCI)*, 2020, pp. 184–189.

[30] Z. Long and W. Jinsong, "Network traffic classification based on a deep learning approach using netflow data," *Comput J*, p. bxac049, 2022.

[31] Z. Wu, Y. Dong, X. Qiu, and J. Jin, "Online multimedia traffic classification from the QoS perspective using deep learning," *Computer Networks*, vol. 204, p. 108716, 2022.

[32] W. Wei, H. Gu, W. Deng, Z. Xiao, and X. Ren, "ABL-TC: A lightweight design for network traffic classification empowered by deep learning," *Neurocomputing*, vol. 489, pp. 333–344, 2022.

[33] A. I. Owusu and A. Nayak, "A framework for QoS-based routing in SDNs using deep learning," in *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, 2020, pp. 1–6.

[34] Z. Wu, Y. Dong, X. Qiu, and J. Jin, "Online multimedia traffic classification from the QoS perspective using deep learning," *Computer Networks*, vol. 204, p. 108716, 2022.

[35] A. Wani and R. Khaliq, "SDN-based intrusion detection system for IoT using deep learning classifier (IDSIoT-SDL)," *CAAI Trans Intell Technol*, vol. 6, no. 3, pp. 281–290, 2021.

[36] R. Batra, V. K. Shrivastava, and A. K. Goel, "Anomaly Detection over SDN Using Machine Learning and Deep Learning for Securing Smart City," in *Green Internet of Things for Smart Cities*, CRC Press, 2021, pp. 191–204.

[37] M. R. Hadi and A. S. Mohammed, "A novel approach to network intrusion detection system using deep learning for Sdn: Futuristic approach," *arXiv preprint arXiv:2208.02094*, 2022.

[38] B. Bağiröz, M. Güzel, U. Yavanoğlu, and S. Özdemir, "QoS Prediction Methods in IoT A Survey," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 2128–2133.

[39] CiscoPress, "Network Fundamentals: Introduction to Network Performance Measurement." 2022.