A Software-Defined Queuing Framework for QoS Provisioning in 5G and Beyond Mobile Systems

Aiman Nait Abbou, Tarik Taleb, and JaeSeung Song

***Abstract*—There is an ever-increasing demand for network technologies supporting Ultra-Reliable Low Latency Communications (URLLC) services and their co-existence with best-effort traffic. By way of example, reference can be made to the emerging 5G mobile networks. In this vein, this paper investigates the Software-Defined Networking (SDN) technology capabilities for providing Quality of Service (QoS) guarantees. Specifically, we present a testbed, under development, dubbed Software-Defined Queueing (SDQ). This framework leverages QoS provision functionalities of SDN. SDQ can be regarded as a framework for testing traffic engineering solutions in networks with deterministic QoS support. By using SDQ, we develop and test a specific solution that chooses the optimal queue and path for every incoming flow in order to reduce the workload imbalances in the network. For the experimental setup, we consider a generic SDN network whose bridges include three priority queues at every output port. Furthermore, we compare the aforementioned solution with a best-effort network and an SDN-enabled network with QoS support configured by default. The obtained results show that the envisioned solution outperforms the baseline one of SDN and the best-effort solution in terms of the average latency recorded.**

***Index Terms*—SDN, QoS, Deterministic Networking, TSN, URLLC, 5G and beyond.**

I. Introduction

Best-effort networking technologies, including Ethernet, are not suitable for time-sensitive applications such as industrial robotic control and remote automotive control. These industries have been forced to develop their own digital solutions using dedicated communication infrastructures. Nevertheless, none of these infrastructures were satisfying due to the incurred deployment costs, lack of flexibility and adaptability, and lack of real-time configuration capabilities [1].

The recent evolution of mobile technologies has shifted the efforts towards developing more efficient mobile network (MN) systems, capable of sustaining ultra-low latency and highly reliable connectivity. The fifth generation of mobile communications systems, 5G, is a noticeable example. With the 5G technology, mobile operators are expected to guarantee Quality of Service (QoS) with different requirements (e.g. bounded-latency, ultra-reliability, and zero data loss). Effectively, 5G is expected to support a wide library of services, that can be categorized into three major areas, namely Ultra-Reliable Low Latency Communication (URLLC) use cases, Massive Machine Type Communication (mMTC) use cases, and Enhanced Mobile Broadband (eMBB) use cases. As the name infers, the services of the former category highly depend on the network latency. However, the main problem resides in the nature of the underlying networking protocols. For example, the Transmission Control Protocol (TCP) is not optimal for the timely delivery of packets [2]. This has made the efforts to minimize the latency and offer a better QoS for TCP-based time-sensitive applications highly challenging.

To overcome the limitations of TCP, Deterministic Networking (DetNet) has been proposed. The main goal of DetNet is a converged network, including the convergence of sensitive non-IP networks onto a common network infrastructure [3]. In other words, DetNet aims to allow the co-existence of all types of services (e.g., critical and best-effort) over a programmable and cost-effective Ethernet-based forwarding plane. The main enablers of DetNet are Software Defined Networking (SDN) and Time Sensitive Networking (TSNs). The latter is an Ethernet-based technology that can provide deterministic performance guarantees up to Layer 2. The queuing algorithms defined in TSN standards offer many potentially-configurable parameters. This flexibility can be exploited through the SDN concept.

SDN is able to provide network services with real-time programmability [4]. An SDN controller who has a global view of the network deals with the decision making and communicates with Packet Data forwarding devices, this process is done through a Southbound Application Programming Interface (API) such as Openflow [5]. This separation between the data plane and control plane has gotten the attention of researchers to improve the QoS provisioning of today's various network applications. QoS can take advantage of SDN in various ways, such as routing mechanisms, network monitoring, resource reservation, queue management, and scheduling mechanisms [6].

To support strict end-to-end (E2E) QoS for various time-sensitive and/or bandwidth-intensive services, it is important to be able to manage the traffic of these services, not only by directing them via the right routers and switches but also by indicating to them the right queue they should traverse at each network node. A fine-granular SDN paradigm at the queue management level becomes then mandatory. In this regard, this paper introduces our Software-Defined Queuing (SDQ) framework, a testbed under development at MOSA!C Lab [7].

SDQ considers a fully programmable network by defining queueing algorithms and managing the priority queues on the data forwarding devices. This prototype is based on two main modules. The first one manages the queues based on the

A. Nait Abbou and T. Taleb are with Aalto University, Espoo, Finland (e-mail: firstname.lastname@aalto.fi)

T. Taleb is also with Sejong University, Seoul, South Korea and with the University of Oulu, Oulu, Finland

J. Song is with Sejong University, Seoul, South Korea (e-mail: jssong@sejong.ac.kr)

incoming traffic. The second module makes traffic engineering tasks by finding the optimal path and balancing the load. The final goal of the SDQ framework is to provide an experimentation environment to test the different candidate network solutions for realizing the transport networks of B5G and 6G and assist the design, development, and evaluation of Artificial Intelligence (AI)-based solutions for queue allocation for each flow type. In this paper, we will focus on describing the configurations required by SDQ to provide QoS support in SDN-enabled networks. Moreover, we include some preliminary results validating the proper operations of the described SDQ setup.

The remainder of this paper is organized as follows. Section II discusses some related research work. Section III describes the prototype and highlights some potential use cases. Section IV presents the experimental results. Finally, Section V concludes the paper.

II. Related Works

In [8], Tomovic *et al.* proposed a SDN control framework for QoS provisioning. The authors implement their solution on top of a Python-based open-source SDN Controller (POX) with OpenFlow v1.0. This framework relies on Linux built-in Hierarchical Token Bucket (HTB) for regulating the traffic. It creates a dedicated queue for every priority flow at every output interface. The performance evaluation compared the proposed solution with best-effort and Integrated service (IntServ). Unlike this work that focuses on how SDN can improve the throughput compared to best-effort or IntServ, our approach envisions scenarios whereby existing queues are saturated (or simply not suitable) and there is a need to create and manage new queues to accommodate new traffic with stricter E2E latency requirements.

Goto *et al.* [9] introduced a Deterministic Service (DetServ) framework with two models based on network calculus theory. The first model, called Multi-Hop Model (MHM), assigns a rate and a buffer budget to each queue in the network, while the second model dubbed as Threshold-Based Model (TBM), specifies a maximum delay for each queue in order to guarantee the service reliability. In comparison to this work, the proposed SDQ framework focuses on the E2E latency in scenarios with filled or not suitable queues. Dutra *et al.* [10] proposed a solution that allows for each flow an E2E QoS based on the queue support in OpenFlow. Unlike SDQ, this work focused more on improving resource utilization and reducing the incurred cost, and that is by limiting the number of switches used for transmissions. Moreover, the multi-path approach is used to ensure an efficient resource allocation considering service demands.

YAN *et al.* [5] proposed HiQoS, an SDN based QoS solution. HiQoS guarantees a certain degree of availability as it exploits multiple paths between the sender and the receiver; the bandwidth can be assured through queuing algorithms for each traffic type. The performance evaluation of HiQoS shows its superiority, compared to a single path QoS solution with and without differentiated services (LiQoS and MiQoS, respectively). Yet, HiQoS was experimented with using mininet and not a real scenario. Furthermore, SDQ considers balancing the load instead of a straight multi-path algorithm. Moreover, the queues and the flow rules are created depending on the real-time dynamics of the network.

Shu *et al.* [11] proposed a novel QoS framework for network slicing based on SDN and Network Function Virtualization (NFV) for 5G and beyond services. The framework considers three SDN controllers; each one manages part of the network, namely Radio Access Network (RAN), Transport Network (TN), and Core Network (CN). The E2E decisions follow a hop-by-hop mode whereby the SDN controllers collaborate with each other. The prototype implements Open Network Operating System (ONOS) as an SDN controller and mininet to emulate a network environment. The results of the simulations show the effectiveness of the proposed QoS framework on scheduling network resources for various network slices and provide reliable E2E connection service for users according to pre-configured QoS requirements. The authors of this paper have targeted the issue of establishing QoS-sensitive network slices in a 5G network. The envisioned SDQ framework can complement this work by providing the necessary instructions to autonomously create queues, define flow rules without human intervention, and communicate them to the forwarding devices.

All the above-mentioned studies adopted experimental and prototyping approaches to evaluate the performance of their respective solutions. Goto *et al.* [12] took a different path as they analytically evaluated the performance of SDN (more specifically OpenFlow–based) networks. They suggested a queuing model to analyze an OpenFlow–based SDN network as realistically and accurately as possible. This approach also considers a classification of the incoming packets to a switch from the controller.

III. Software Defined Queueing (SDQ)

A. Use cases

Providing QoS is becoming an important aspect of any network architecture. The new emerging applications and services have different specifications. Industrial Internet of Things (IIoT) networks such as smart factories have strict requirements in terms of ultra-low delay, high reliability, and flexible dynamic configuration. For instance, performing real-time tasks on a millisecond scale in cycle time is highly challenging. As a result, research on URLLC has drawn a lot of attention in recent years.

TSN has been developed to address these requirements, but it is still far from maturity. Effectively, TSN depends heavily on time-synchronization [13], and any unpredictable event can disrupt the communication. Furthermore, it lacks flexibility and is known for its inefficient usage of the network resources, which make TSN-based solutions far from being cost-effective. Additionally, they do not support best-effort services. In contrast, the QoS research community aims to support reliable deterministic services alongside best-effort services, without the need for any special equipment or major upgrades in the existing infrastructure.

In Industrial Internet, the production is customized for each service and the requirements are listed as per the order of

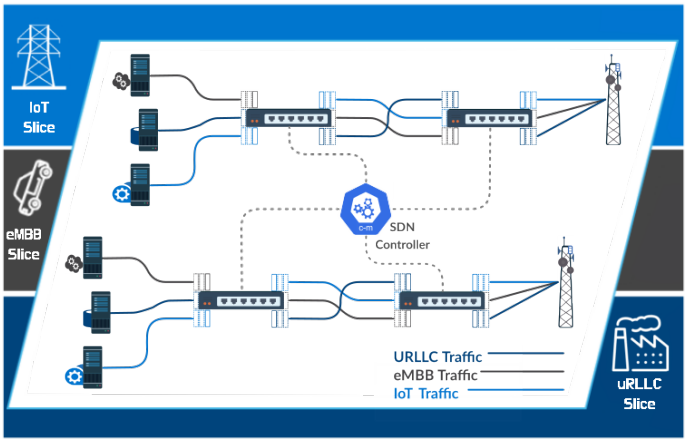


Fig. 1: 5G physical network divided into several network slices with different QoS requirements.

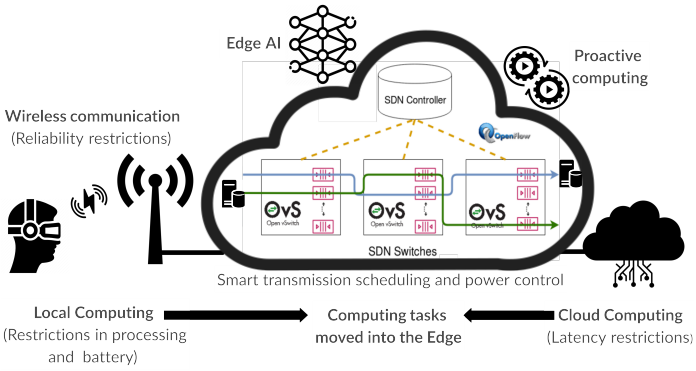


Fig. 2: SDQ combination with AI and Edge computing for URLLC use case (XR)

their priority. Moreover, deterministic services are a must for communications either intra-subfactory from sensors to actuators or inter-subfactories. SDQ framework is developed to accomplish these requirements. It is a centralized traffic engineering framework, built for the real-time guarantees of deterministic latencies, by transferring the intelligence of the network to an SDN controller with a global view of the network. The controller can dynamically apply more complex network resource management policies and mechanisms such as dynamic optimal path routing or network reconfiguration based on real network conditions [14]. In this framework, the SDN controller works alongside a management protocol to configure the forwarding devices according to the requirements of the services and their dynamics. SDQ is able to create, on-demand and in real-time, a maximum of 65,000 queues on each port with different priorities and guaranteed bandwidths on multiple switches. This can be massive especially for Industrial networks with long-distance communication or multiple hops. The main objective of SDQ is to ensure that the E2E communication path, formed from the switches traversed and even the queues within each switch, can accomplish the expected E2E latency requirements while still providing best-effort services. Figure 1 illustrates SDQ combination with a 5G Network. Depending on the use case, we have several network slices sharing the same physical infrastructure [15]. Moreover, the traffic of each slice goes via different groups of queues within each switch.

Another use case that can take advantage of SDQ framework is Extended Reality (XR). Technically XR sits in the middle of URLLC and eMBB. Moreover, XR applications need a highly precise latency and high-reliability demands depending on the expected feedback (i.e., visual, visual, and haptic). Figure 2 breaks down a potential scenario of SDQ and XR applications. First, due to the limitations of battery and processing power capabilities, the computation duties are shifted from the data source to the edge cloud. In this particular use case, AI can be leveraged to forecast the field of view and the user's location to minimize the latency, SDQ can be exploited to ensure bounded and low E2E latency from the data source to the edge. In order to achieve these strict delay requirements, SDQ selects the right combination of path/port/QoS/queue to ensure the E2E latency for that service.

B. Prototype

SDQ is an application installed on top of ONOS. It contains two main modules, namely queue management, and traffic engineering. The former utilizes ONOS Command Line Interface (CLI) and the Open vSwitch Database Management (OVSDB) Protocol to manage queues at the switches while the latter decides on the flow rules to communicate to switches. Based on these rules, flows are being handled by having their packets traverse designated queues at each switch along the communication path. Figure 3 (A) illustrates the high-level architecture of the SDQ framework.

In SDQ, for every flow newly coming into the network, a packet-in message is generated by the respective access node (i.e., edge bridges of the SDN forwarding plane). In response, the process shown in Figure 3 (B) is triggered. In the remainder of this section, we describe the main operations of the two SDQ modules.

- **Traffic engineering**: This module is responsible for choosing an E2E path for every incoming flow. Firstly, it checks if there is already an allocated path (flow rule) for the incoming packet. Otherwise, it lists all available paths between the source and the target destination and computes a weight for each of them as follows:

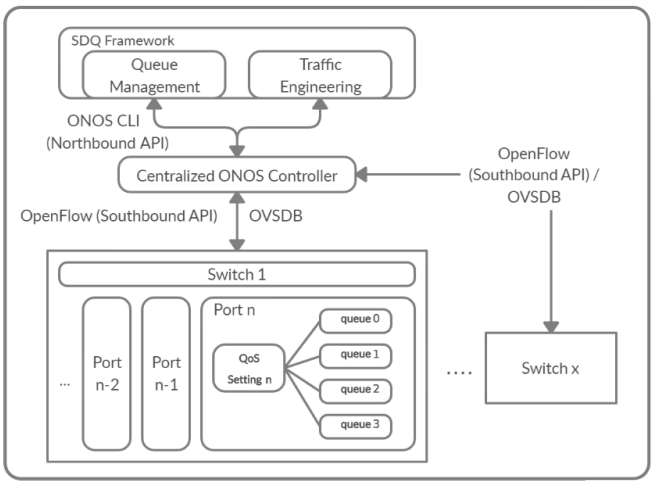
$$Weight = 100 - \left(100 * \frac{Available_{BW}}{Maximum_{BW}}\right) \quad (1)$$

whereby $Maximum_{BW}$ and $Available_{BW}$ denote the maximum and available E2E capacity for a given path, respectively. The $Available_{BW}$, in turn, can be computed as:

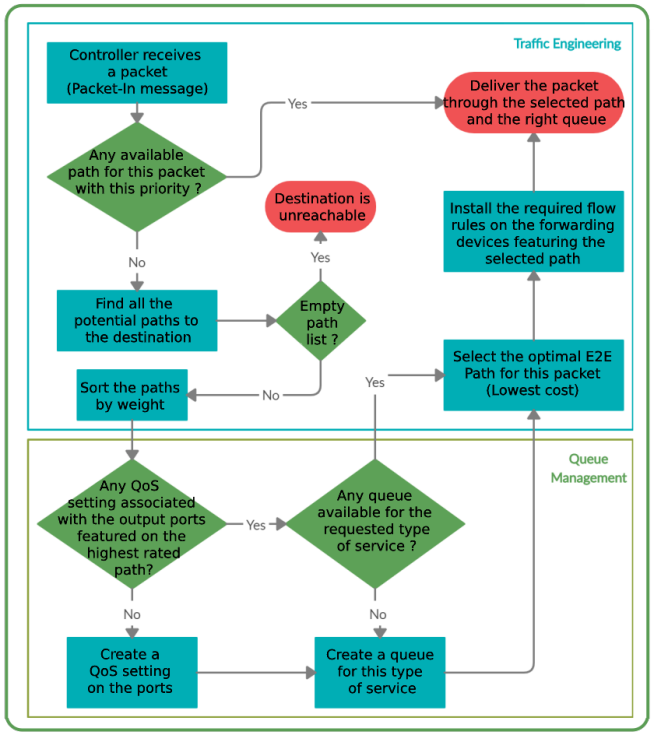
$$Available_{BW} = Maximum_{BW} - Occupied_{BW} \quad (2)$$

whereby $Occupied_{BW}$ is the bandwidth already reserved along the path for the ongoing flows.

As can be observed from Equation (1), the weight value computed for every candidate path ranges from 0 to 100, where 100 represents a path fully loaded (i.e., there is at least one link in the path whose utilization is 100%). Indeed, the traffic engineering algorithm will choose the path with the smallest



(A) SDQ framework architectural model



(B) SDQ Workflow

Fig. 3: SDQ components and functionalities

weight or, in other words, the least loaded one. In this way, the algorithm reduces the load imbalances in the network. The chosen path (i.e., that with the lowest weight) is used as a reference to install the QoS settings and the required queues. The data forwarding devices along the selected path are then communicated the necessary flow rules according to which packets of the target flow will be handled and that is using the queues designated for the flow.

- **Queue Management**: For each flow, this module enforces the QoS settings, received from the SDQ controller, on the forwarding devices along the communication path selected for the flow. This module implements the different queue management functions, such as creating queues, modifying queues, deleting queues, and specifying the scheduling algorithms, and setting the parameters for these queues. The detailed workflow is described in Figure 3 (B).

IV. Performence Evaluation

A. Technical details

In order to evaluate the performance of SDQ, we consider a network composed of three virtual switches as depicted in Figure 4. Each SDN switch has a non-preemptive strict priority scheduler at every output port. We use ONOS as an SDN controller, and OpenVSwitch (OVS) to implement the SDN bridges. Table I includes all technical details of the envisioned emulation setup.

In the performance evaluation, the following four scenarios are considered and that is as shown in Figure 4.

- **Scenario 1 (HighPriority):** Host 1 generates a total of 20,000 High Priority (HP) packets (traffic of interest). Host 2 generates the same amount of packets as H1 but it is subdivided between a Medium and Low priority (MP and LP) packets (interfering traffic).
- **Scenario 2 (MediumPriority):** Host 1 generates MP packets. Host 2 generates HP and LP packets.
- **Scenario 3 (LowPriority):** Host 1 generates LP packets. Host 2 generates MP and HP packets.
- **Scenario 4 (Best-Effort):** Host 1 generates 20,000 packets without any specific priority as a traffic of interest, and Host 2 does the same with the flow of interference.

For each queue, SDQ can allocate a minimum rate and a ceiling (maximum) rate. In the envisioned scenarios, the HP queue (htb 1:2) will have at least 80% of the available bandwidth if the output port has LP or MP packets on the lower queues. Otherwise, the HP queue can take advantage of the full bandwidth. For the medium queue (htb 1:3), it depends on whether the HP packets exist or not. In the envisioned scenarios, we give it a minimum of 10%; this means that in the worst-case scenario while competing with the HP queue, the MP queue will still get at least 10% of the bandwidth, the

TABLE I: Technical details of the emulation setup.

Operating System (OS)	Ubuntu 19.04
Softwares and protocols	ONOS 2.3, OpenVSwitch 2.11.0, Nping 0.7.70, OpenFlow 1.4,
Topology	Linear
Packets generated	Total of 40,000 Packets divided between Host 1 and Host 2 (20,000 packets each)
Generation rate	1000 packet/second
Packet size	162 bytes
Flows	Flow of Interest (H1) and flow of interference (H2)
Priority queues	High, Medium, Low and best-effort.
Bandwidth	Default 10Gbps on each OVS port
QoS Maximum rates	S1-eth4 : 1Mbps, S1-eth2 and S2-eth2 : 10Mbps
Queues minimum rate	HP htb 1:2 (80% of the QoS maximum rate), MP htb 1:3 (10%), LP htb 1:4 (7.5%) and best-effort htb 1:1 (2.5%)
Queuing mechanism	Hierarchical Token Bucket (HTB)

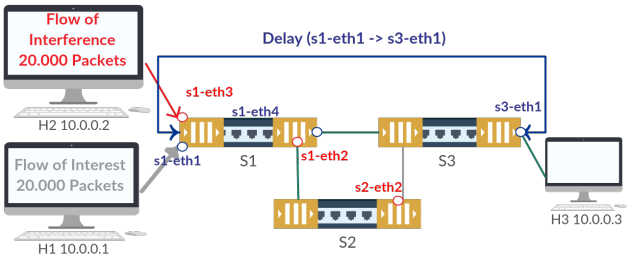
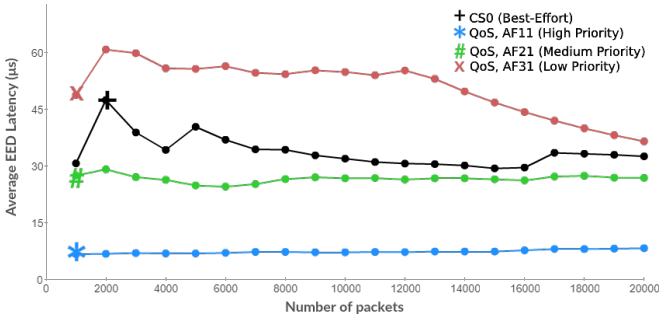
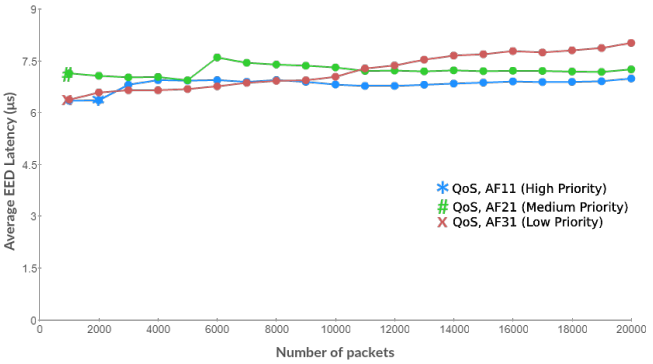


Fig. 4: Network topology illustrating the envisioned scenarios.



(A) The average latency experienced by the packets handled by SDN-based QoS enforcement solution



(B) The average latency of the packets handled by SDQ framework

Fig. 5: Experiment results of each solution

LP queue (htb 1:4) has a minimum rate of 3.8% of the total bandwidth, and all queues can have the full bandwidth if the higher queues are empty. This approach can help us visualize and study the network performance under realistic conditions whereby we stress the switches (especially Switch 1) with a large number of packets and ensure none of the queues is empty.

B. Experimental Results

In the performance evaluation, since we target low latency communications, we consider the network latency as the main comparison metric. Figure 5 depicts the obtained results.

Figure 5 (A) shows the average latency experienced by packets of best-effort traffic as well as of traffic handled by an SDN-based QoS enforcement solution. In both experiments, we adopted a reactive forwarding based on the shortest path selection metric. As expected, the HP flow has the lowest latency followed by the MP flow. The best-effort traffic actually experiences a shorter latency than the LP traffic. This is mainly due to the fact that packets of best-effort traffic and LP traffic are considered the same, and are forwarded in a FIFO fashion exploiting the full available bandwidth (1Mbps in this case). From Figure 5 (A), we can also notice that all packets of the HP and MP traffic experience more or less the same latency whereas the latency experienced by packets of best-effort traffic varies significantly. Indeed, it fluctuates for the first 1000 packets and then converges to an average value afterward (i.e., from the 1000th packet and onward). It is also noticed that the latency experienced by LP packets decreases after the 12,000th packets. This is mainly since more bandwidth becomes available.

It is worth mentioning that when SDQ is not in use, packets of all traffic types traverse the shortest path (i.e., from Switch 1 to Switch 3 as in the envisioned network topology). However, even though both ports s1-eth2 and s1-eth4 have the same configured link speeds, the QoS setting configured on the ports is different. Indeed, the maximum rate of s1-eth4 is 1Mbps and that of s1-eth2 is 10Mbps. This intuitively makes the long path (Switch 1 to Switch 2 and then to Switch 3) more attractive in terms of performance. The SDQ framework exploits this metric and selects the long path through switch 2. From Figure 5 (B), we observe that the average latency of the three types of services are comparable. Indeed, exploiting the high bandwidth offered through Switch 2, SDQ could handle incoming packets at each switch faster and without significant queuing delays. It shall be also noticed that the latency experienced by packets of the same traffic category does not fluctuate frequently and remains stable below a target value, which is in line with the core spirit of deterministic networking.

V. Conclusion

This paper introduces our envisioned SDQ framework. For each flow, the core idea beneath SDQ is to explore all available E2E paths and to select the optimal one in terms of both QoS performance (i.e., E2E latency) and load distribution. For each selected path, SDQ also makes queue management in real-time by creating new queues, deleting or updating existing ones as per the deterministic networking needs of the target flow. The performed experiments have shown a stable, bounded, and very low latency for packets of high and medium priority traffic. Whilst in the conducted experiments, the selection of communication paths and queues was done manually, our future research work will focus on the automation of this selection leveraging artificial intelligence techniques.

Acknowledgment

This work was partially supported by the European Unions Horizon 2020 Research and Innovation Program through the MonB5G Project under Grant No. 871780. It was also supported in part by the Academy of Finland 6Genesis project under Grant No. 318927 and by the Academy of Finland CSN project under Grant No. 311654. Prof. Song was supported by the Korea Research Foundation with the funding of the Ministry of Science, Technology, Information and Communication under Grant No.2018-0-88457.

REFERENCES

[1] H. H. Houh, P. Anderson, and C. Gadda, "Telecommunication method for ensuring on-time delivery of packets containing time-sensitive data," Nov. 22 2005, uS Patent 6,967,963.

[2] J. Luo, J. Jin, and F. Shan, "Standardization of low-latency tcp with explicit congestion notification: A survey," *IEEE Internet Computing*, vol. 21, no. 1, pp. 48–55, 2017.

[3] N. Finn, P. Thubert, B. Varga, and J. Farkas, "Deterministic Networking Architecture," Internet Engineering Task Force, Internet-Draft draft-ietf-detnet-architecture-13, May 2019, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-detnet-architecture-13

[4] R. A. Addad, T. Taleb, H. Flinck, M. Bagaa, and D. Dutra, "Network slice mobility in next generation mobile systems: Challenges and potential solutions," *IEEE Network*, vol. 34, no. 1, pp. 84–93, 2020.

[5] J. Yan, H. Zhang, Q. Shuai, B. Liu, and X. Guo, "Hiqos: An sdn-based multipath qos solution," *China Communications*, vol. 12, no. 5, pp. 123–133, May 2015.

[6] M. Karakus and A. Durresi, "Quality of service (qos) in software defined networking (sdn): A survey," *Journal of Network and Computer Applications*, vol. 80, pp. 200–218, 2017.

[7] "MOSA!C Lab mobile softwarization & service customization," http://www.mosaic-lab.org/, accessed: 2020-01-07.

[8] S. Tomovic, N. Prasad, and I. Radusinovic, "Sdn control framework for qos provisioning," in *2014 22nd Telecommunications Forum Telfor (TELFOR)*, Nov 2014, pp. 111–114.

[9] J. W. Guck, A. Van Bemten, and W. Kellerer, "Detserv: Network models for real-time qos provisioning in sdn-based industrial environments," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1003–1017, Dec 2017.

[10] D. L. C. Dutra, M. Bagaa, T. Taleb, and K. Samdanis, "Ensuring end-to-end qos based on multi-paths routing using sdn technology," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

[11] Z. Shu and T. Taleb, "A novel qos framework for network slicing in 5g and beyond networks based on sdn and nfv," *IEEE Network*, 2020.

[12] Y. Goto, B. Ng, W. K. Seah, and Y. Takahashi, "Queueing analysis of software defined network with realistic openflow–based switch model," *Computer Networks*, vol. 164, p. 106892, 2019.

[13] J. Prados-Garzon, T. Taleb, and M. Bagaa, "Learnet: Reinforcement learning based flow scheduling for asynchronous deterministic networks," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[14] T. Taleb, I. Afolabi, and M. Bagaa, "Orchestrating 5g network slices to support industrial internet and to shape next-generation smart factories," *IEEE Network*, vol. 33, no. 4, pp. 146–154, 2019.

[15] T. Taleb, I. Afolabi, K. Samdanis, and F. Z. Yousaf, "On multi-domain network slicing orchestration architecture and federated resource control," *IEEE Network*, vol. 33, no. 5, pp. 242–252, 2019.