SDN Framework for QoS provisioning and latency guarantee in 5G and beyond

Sofiane MESSAOUDI
EURECOM
Sophia Antipolis, France
sofiane.messaoudi@eurecom.fr

Adlen Ksentini
EURECOM
Sophia Antipolis, France
adlen.ksentini@eurecom.fr

Christian BONNET
EURECOM
Sophia Antipolis, France
christian.bonnet@eurecom.fr

***Abstract*—In this paper, we unveil the Software-Defined Low Latency (SDLL) framework based on Software-Defined Networking (SDN) to provision the Quality of Service (QoS) and guarantee ultra-low latency in 5G and beyond Transport Networks (TN). SDLL aims to tackle Time Sensitive Networking (TSN)'s weaknesses by providing agility and flexibility in terms of Traffic Engineering (TE) and Queue Management (QM) to guarantee low end-to-end (E2E) latency even under congested links. SDLL provides a flexible and on-demand way to change end-to-end paths and queue configurations (ex., add or remove queues). We conducted extensive experimentation by implementing SDLL using Open Network Operating System (ONOS) and Open vSwitch (OVS) tools and comparing its performances against two standard solutions: SDN Shortest Path (SDNSP) (one queue per port) and Software-Defined QoS (SDQoS) (three queues per port). Obtained results indicate that SDLL can guarantee low E2E latency compared to the two other solutions, particularly when: (1) the links are congested and (2) many low-latency critical services are run in parallel.**

I. Introduction

Although 5^{th} Generation (5G) is still under deployment, the research community is already establishing the requirements of 6^{th} Generation (6G) systems. Ultra low latency is one of the main requirements that should be natively supported, which is not been well addressed in 5G. The expected services and applications [1] in 6G such as holograms and tactile Internet need a highly synchronized communication from the Radio Access Network (RAN) to the network service to guarantee ultra-low End-To-End (E2E) latency.

5G and beyond connectivity involves different network segments, i.e., RAN, Core Network (CN), and Transport Network (TN). The TN includes not only the Backhaul links connecting the CN to the RAN but also the Fronthaul and Midhaul, which result from the functional splits of the RAN that appeared in 5G [2]. Therefore, to guarantee a very low E2E latency, the three parts need to be optimized, i.e., RAN, CN, and TN. For the RAN, works need to be done to reduce the slot duration further and improve the Hybrid Automatic Repeat Request (HARQ) mechanisms [3]. The reduction of latency at the CN has already been addressed in 5G, where network slicing is used. However, ensuring very low latency at the TN level is still challenging.

Most of the existing approaches to guarantee low latency in TNs are relying on Time Sensitive Networking (TSN), which was devised by the Institute of Electrical and Electronics Engineers (IEEE). TSN specifications define two standards: The first one is based on synchronous communication (IEEE 802.1Qch and IEEE 802.1Qbv) [4]. It uses a precise and common time reference shared among all the TSN devices (i.e., routers). It allows the scheduling of the traffic transmission in a deterministic way to maintain a lower bound for the latency. However, it needs a network-wide coordinated time, which hinders the scalability of the network. In addition, using reserved time slots for each flow may lead to poor utilization of network resources. The second one is based on asynchronous communications (IEEE 802.1Q) [5]. In contrast to the synchronised version, no common time reference is needed and shared between devices. Each device implements advanced scheduling on a two-level queuing hierarchy to shape the traffic and achieve a bounded latency at each hop.

Although the TSN is an interesting approach, it still has weaknesses in fully sustaining the low latency expectation of 5G and beyond services. On one hand, TSN provides hop-by-hop bounded latency and not E2E latency guarantee; the latter is highly needed when multiple paths are available. Indeed, high E2E visibility is crucial when the links get congested, and changing the path is necessary to continue ensuring low latency. On the other hand, TSN adopts a static configuration of the different queues of the switch, which has limitations when several Ultra-Reliable Low Latency Communications (URLLC) traffics are competing for the same high-priority queue. Indeed, 5G specifications define more than 10 services as delay critical, with different levels of criticality. For instance, Electricity Distribution is the highest delay critical service (5ms E2E latency), while Vehicle-to-Everything (V2X) message and real-time gaming can be considered less critical (50ms E2E latency). Accordingly, the network should be in the capacity to protect the highest delay critical services, even in the presence of others. This can be achieved only by using an agile Queue Management (QM) system, which is impossible in TSN.

To overcome the above-cited limitations, in this paper, we embrace an Software-Defined Networking (SDN)-based approach. SDN provides all the flexibility and agility to guarantee E2E low latency even when several delay-critical services are running in parallel. SDN provides the necessary functions, known as Create, Read, Update, and Delete (CRUD), to program and monitor the network elements. It includes dynamic QM, to ensure a real-time E2E control of the different flows running in the network. The proposed solution, namely (Software-Defined Low Latency (SDLL)), relies on the new data plane 5G specifications. It uses the Quality Flow Identification (QFI) [6] information to map the 5G services to TN traffic classes (and queue) to fulfill the needed Quality of Service (QoS). Furthermore, SDLL adapts to the network load by dynamically updating queues (add or remove) which permits segregating among URLLC traffics and hence protecting the highest priority traffics (strongest delay-critical services). The contributions of this work are manifolds:

- We introduce a SDN-based novel framework for QoS provisioning in 5G and beyond, featuring a low latency guarantee;
- We propose a Traffic Engineering (TE) algorithm that ensures efficient resource management while providing a bounded E2E latency for URLLC services even in a loaded network;
- We present a dynamic QM algorithm to enforce priority among URLLC services.

The rest of the paper is organized as follows: Section 2 introduces the state-of-the art solutions. Our solution is shown in Section 3 and evaluated in Section 4. We conclude the paper in Section 5.

II. Related Work

In this section, we review different works that have been done in correlation with QoS provisioning and latency guarantee in communication networks.

In [7], authors developed an SDN framework named Software-Defined Queuing (SDQ) for QoS provisioning by selecting the right path and queue according to the network bandwidth management. Indeed, they route the traffic through the path with the smallest available bandwidth weight. However, their configuration was static and not updated compared to our SDLL solution.

[8] proposed a queuing model to analyze an OpenFlow–based SDN network. They also considered a classification of the incoming packets. But, the topology was fixed and included only one switch, which reduced the number of available paths and hence the complexity of the conducted analysis.

[9] delivered a framework that synthesizes paths through the network. To guarantee that flows meet both the bandwidth and E2E timing requirements, they solved a multi-constraint optimization problem using a heuristic algorithm and exhaustive emulations and experiments on hardware switches to demonstrate the techniques and feasibility of their approach based on SDN. Despite that, their solution is theoretical and is not dynamic in term of queue management comparing to ours.

In [10], evaluations were done on the performance of LEARNET through simulation in a 5G asynchronous deterministic backhaul network where incoming flows have characteristics similar to the four critical 5G QoS Identifiers (5QIs) defined in 3rd Generation Partnership Project (3GPP) Technical Specification [11]. This solution is based on asynchronous TSN which provides hop-by-hop bounded latency and not E2E latency guarantee (the latter is highly needed when multiple paths are available).

All the above solutions do not consider the QM part and their monitoring. Instead, our solution considers several inputs (i.e., packets priority, number of hops, number of queues, and their loads) in the path computation to route traffic, aiming at achieving a guaranteed E2E low latency for critical applications even in a loaded network.

III. Software Defined Low-Latency (SDLL)

A. SDLL concept and interaction with 5G data plane

SDLL is a SDN-based solution that controls, monitors, and manages a 5G TN (Backhaul) composed of a number of programmable switches. SDLL uses the CRUD functions to ensure agility in managing 5G service traffic to guarantee required QoS, and particularly low-latency for delay-critical services (known as URLLC services). Figure 1 shows the positioning of SDLL in a 5G data plane. To recall, to create a communication link with outside and start sending and receiving traffics, a User Equipment (UE) needs to establish a session that creates dedicated bi-directional tunnels or bearers. At the core network side, the bearer encapsulates UE's packets Internet Protocol (IP) in GPRS Tunneling Protocol (GTP) tunnels between the gNodeB (gNB) and User Plane Function (UPF). The GTP header includes information on the user traffic, like the QoS that the gNB and UPF should apply to the tunnel. This information in 5G corresponds to the 5G QFI field. 5G QoS Identifier (5QI) is a mechanism whereby packets are classified into various QoS classes. Thus, the QoS can be tailored to specific requirements. Each QoS class has specific QoS characteristics (such as packet delay and packet loss). There are approximately two dozen standard 5QI values, grouped into two types of resources: Guaranteed Bit Rate (GBR) and Non-Guaranteed Bit Rate (Non-GBR). QoS characteristics include resource type, priority level (lower number implies higher priority), Packet Delay Budget (PDB), Packet Error Rate (PER), Maximum Data Burst Volume (MDBV), etc. The QFI value is assigned by the 5G CN according to the UE subscription and the service to run. Therefore, QFI is a critical value that needs to be taken into account when carrying the tunnel traffic over the 5G TN or Backhaul.

As depicted in the figure, the SDLL is composed of the SDN controller and the SDN applications that run the TE and QM. The SDN control plane is independent of the 5G network infrastructure. The only interaction between SDLL and 5G data plane is the border switches (i.e., the routers connecting the gNB and UPFs to the 5G TN) that need to map the QFI to a Differentiated Services Code Point (DSCP) value. In SDLL, we use the mapping proposed in [12], which specifies a set of 3GPP QoS Class Identifier (QCI) and 5QI to DSCP mappings to reconcile the marking recommendations offered by the 3GPP with the recommendations offered by the Internet Engineering Task Force (IETF). This maintains a consistent QoS treatment between 5G networks and the Internet. It is worth noting that border switches need to parse the GTP header to extract the QFI and modify the IP header (layer 2) with the corresponding DSCP value to route the packet through the 5G TN according to SDLL control. The mapping process is done in the two directions of the traffic, i.e., Uplink and Downlink.

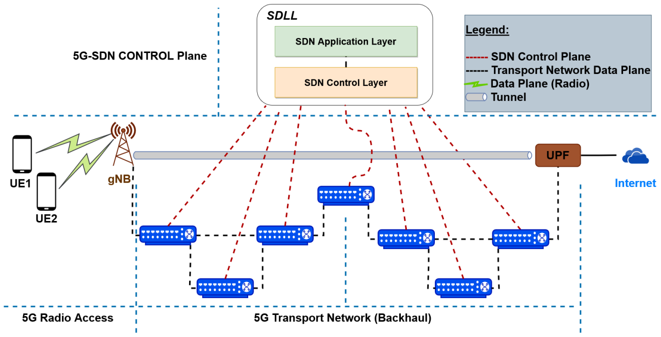


Figure 1: The 5G SDNized network infrastructure.

A more focused representation of SDLL is shown in Figure 2. Here, we detail the different elements interacting SDLL with 5G TN. The solution is represented within the three layers of the SDN. The SDLL application is located on the first layer. It is installed on top of a SDN controller. In this work, it corresponds to a Java-made Open Network Operating System (ONOS) controller. The SDLL applications contain two parts, namely *queue management* and *traffic engineering*. The former utilizes ONOS Representational State Transfer Application Programming Interface (REST API) and the Open vSwitch Database Management (OVSDB) Protocol to manage queues at the switches and get their statistics, while the former decides on the flow rules to communicate to switches via the OpenFlow protocol. Accordingly, each packet entering a switch is routed by designating the right queues along a chosen path. It is worth mentioning that our solution is applicable to Open vSwitch (OVS) but can be intended to any other technology (i.e., switch) with integrated Data Base (DB) and NorthBound Interfaces (NBI) that enable the QM and the flux rules installation.

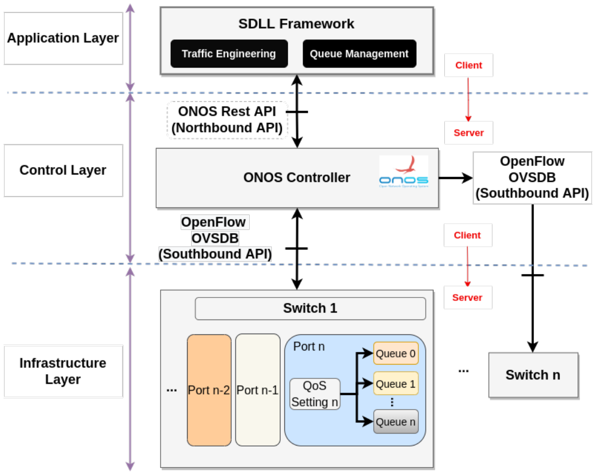


Figure 2: The architecture view of the SDLL framework.

A flow rule ξ is composed of: *Match set* (μ) to identify a flow; *Action set* (τ) to define the actions executed on each packet of the flow; and *Priority* (ρ) that is used to relatively order rules in the forwarding switch. In our solution the Match set includes the DSCP value of the packets, the Action set is to route the packets of each flow through a set of port numbers and queue IDs, while the priority is the same for all the flow rules. Please refer to the equations 1, 2, 3, and 4.

$$\xi = \{[\mu], [\tau], \rho\} \tag{1}$$

$$\mu = \{0, 1, 2, ..., 56\} \tag{2}$$

$$\tau = \{output(port_{number}, queue_{id})\} \tag{3}$$

$$\rho = \{0, 1, 2, ..., 65535\} \tag{4}$$

B. SDLL details

In SDLL, for every flow newly coming into the network, a PACKET-IN message is generated by the respective access node (i.e., border switches of the SDN forwarding plane or Backhaul). As a response, the process shown in Algorithm 1 is triggered. In what follows, we describe the two main components of the SDLL framework along with the workflow they use.

1) Traffic Engineering (TE): This module is responsible for choosing an E2E path for every incoming packet and hence for the 5G flow. It operates as follows: **(1)** Check if there is already an allocated path (i.e., flow rule) for the incoming packet. **(2)** If not, list all the available paths between the source and the target destination. **(3)** Choose the paths according to the packet priority: **-** Shortest path (the path with the minimum number of hops π) for High Priority (HP) packet to satisfy its requirements; **-** The path with the minimum queue number (the sum of the queues number in the ports of a given path) (see Eq. 5) for Medium Priority (MP) packet to be sure that the average output rate of the queues in that path is maximum so that it reduce the E2E transmission delay; **-** The path with the minimum average queue load (the sum of the average queues loads in the ports of a given path) (see Eq. 7) for Low Priority (LP) packets so that the E2E queuing delay be minimal. It is worth mentioning that border switches need to parse the GTP header to extract the QFI and modify the IP header (layer 2) with the corresponding DSCP value to route the packet through the 5G TN.

$$\psi = min\left\{\sum_{i=1}^{\pi-1} \varrho_i\right\}$$
$$\text{where} \left\{\begin{array}{l} \psi \text{ is the minimum queue number in the list of paths,} \\ \pi \text{ is the number of hops between source and destination,} \\ \varrho_i \text{ is the number of queues in a port } i \end{array}\right. \tag{5}$$

$$\gamma = min\left\{\frac{\sigma_{j,i} \times 100}{\delta}\right\}$$
$$\text{where} \left\{\begin{array}{l} \gamma \text{ is the shortest path hop by hop minimum queue load,} \\ \sigma_{j,i} \text{ is the load of queue } j \text{ in port } i, \\ \delta \text{ is the queue size (100 packet for each)} \end{array}\right. \tag{6}$$

$$\varphi = min\left\{\sum_{i=1}^{\pi-1}\sum_{j=1}^{\varrho_i}\left(\frac{\sigma_{j,i} \times 100}{\delta}\right)\right\} \tag{7}$$
$$\text{where} \left\{ \begin{array}{l} \varphi \text{ is the minimum average queue load in all the paths,} \end{array}\right.$$

2) Queue Management (QM): For each flow, this module enforces the QoS settings received from the SDLL framework on the forwarding devices along the communication path selected for the flow. This module implements the different functions: CRUD queues, specifies the scheduling algorithms, and sets the parameters for these queues. The detailed workflow is described in Algorithm 1.

3) SDLL Workflow: The SDLL framework reduces the load imbalances in the network. Also, the chosen path for the specific traffic priority μ ($\{0,1,2,3,4,8,10,12,14\}$ for LP, $\{16,18,20,22,24,26,28,30\}$ for MP and $\{32,34,36,38\}$ for HP) is used as a reference for the QM module to install the QoS settings and the required queues (lines **14**, **22** and **24**). The TE module translates the chosen path into flow rules, which include data such as the output port number and queue Identifier (ID) (line **25**). Furthermore, a threshold is specified for the HP packets by γ with a predicate of (80%, 50% and 30%) of δ. Hence, if γ is exceeded (line **15**), the LP packets are diverted to another φ path. Also, a new queue with higher priority for HP$^+$ packets (μ in $\{40,44,46,48,56\}$) is created in the switch port where the HP packet is located and the other queues in that port will be updated by reducing their maximum rate (lines **17**, **18**). Finally, the flow rules are pushed to the intermediate nodes involved in the communication (lines **25** and **26**).

Algorithm 1 SDLL Workflow

```
1:  Inputs: {Paths} : {Hosts} : {Links} : {Flow Rules} : {Queues statistics}
2:  Outputs: {New Flow Rule with best path and queue}
3:  The controller receives a packet with a $\mu$ (packet-in message)
4:  if it exists a path with a flow rule for this packet with the DSCP $\mu$ then:
5:      Send the packet via this path and queues
6:  else
7:      Find all the potential paths to the destination
8:      if path list is empty then:
9:          The destination is unreachable
10:     else
11:         Sort the list by ($\pi$, $\psi$, $\varphi$)
12:         Check the packet priority $\mu$
13:         if $\mu \geq 32$ then:
14:             Chose the path with the minimum $\pi$
15:             if $\gamma > Threshold$ then:
16:                 Diverts ($\mu < 16$) packets to other $\varphi$ paths
17:                 Create a new queue and update the others maximum rate
18:                 Split the ($\mu \geq 32$) packets and diverts the ($\mu \geq 40$) packets to the new queue
19:             else
20:                 Goto 23
21:         elseif ($\mu < 32$) and ($\mu \geq 16$):
22:             Chose the path with $\psi$
23:         elseif ($\mu < 16$) and ($\mu \geq 0$):
24:             Chose the path with $\varphi$
25:         Install the flow rules on the devices for the path
26:         Goto 5
```

IV. Performance Evaluation

A. Technical Details

In order to evaluate the performance of SDLL framework, we consider a network topology composed of (8) virtual switches and (24) hosts. Figure 3 depicts that topology. We use ONOS as an SDN controller, OVS for the SDN switches and Mininet for the network topology. Table I includes all technical details of the envisioned setup. In the performance evaluation, a maximum of $1,2\times10^6$ packet (5×10^4 packet for each host) have been generated for each of the following two scenarios:

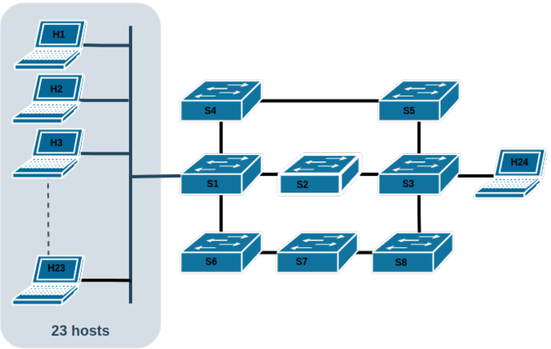


Figure 3: Network topology used for the setup.

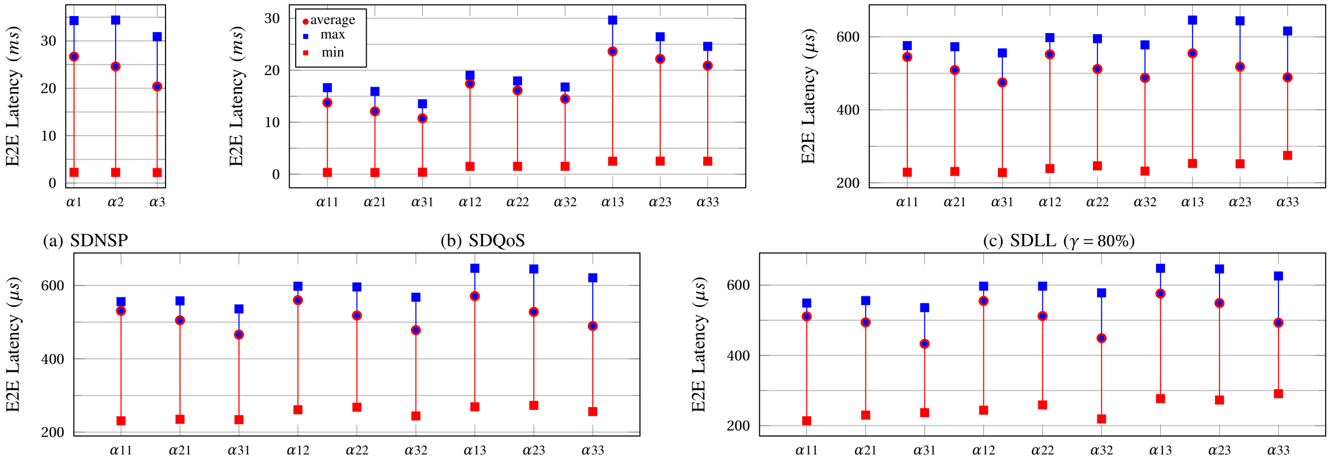
Scenario 1: The 24 hosts are grouped into three sets of 8 hosts. Each group generates the same type of traffic (α_{i1} is HP, α_{i2} is MP, and α_{i3} is LP). In each group, we choose a host as a reference, while the others are seen as interference (i.e., 3 hosts for traffic interests and 21 hosts for traffic interference). We varied the number of packets generated by each host ($\alpha_{1j} = 5\times10^4$, $\alpha_{2j} = 2.5\times10^4$, and $\alpha_{3j} = 5\times10^3$; where j represents the type of the traffic as quoted above), and γ from (80%, 50%, and 30%). Comparison is made with the default and static configurations SDN Shortest Path (SDNSP) and Software-Defined QoS (SDQoS). SDNSP is configured with one queue on each port of the network switches while SDQoS includes three queues on each. Also, there is no QM on the two solutions.

Scenario 2: In this scenario, we focus only on HP traffic. Therefore, 12 hosts generate HP packets and the 12 others HP$^+$ with (5×10^4 packet for each). Only two hosts are considered for traffic interest. The others are seen as interference. We also varied γ from (80%, 50%, and 30%) for HP packets. We compared the results also with SDNSP and SDQoS.

We have put the network under realistic conditions aiming at stressing the switches (especially Switch 1, which is the entry point of the entire hosts) with a large number of packets and ensuring none of the queues are empty.

B. Results

In the performance evaluation, we consider the network latency as the main comparison metric.



(a) SDNSP

(b) SDQoS

(c) SDLL ($\gamma = 80\%$)

(d) SDLL ($\gamma = 50\%$)

(e) SDLL ($\gamma = 30\%$)

Figure 4: Scenario 1 E2E latency results experienced by (HP, MP, LP) packets ($\alpha_{.j}$) and handled by SDNSP and SDQoS static solutions, and SDLL framework with different threshold (γ) and number of packets ($\alpha_{i.}$).

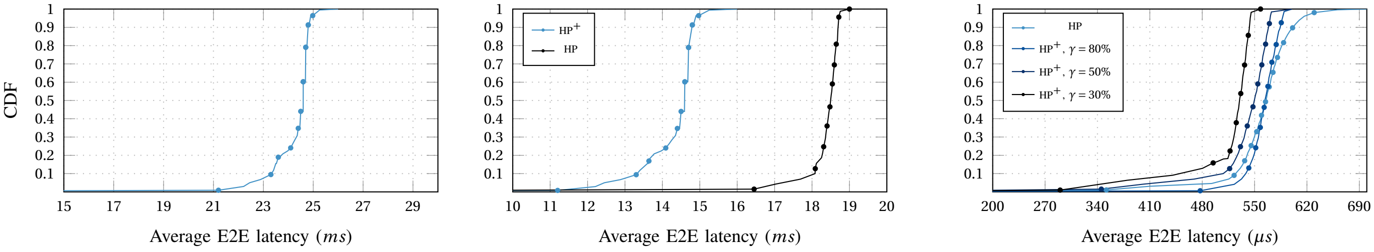
Table I: Technical details of the setup.

Operating System	Ubuntu 20.04.4 LTS
Software and protocols	ONOS 2.7.0, Mininet 2.2.2, OpnVSwitch 2.13.5, Ping iputils s20190709, OpenFlow 1.6
Topology	Linear
Packets generated	Maximum of $1.2 * 10^6$ packet divided between 24 hosts (5×10^4 packet each)
Generation rate	10,000 packet/second
Packet size	65507 bytes
Number of iterations	100
Priority queues	High, medium and low
Bandwidth	50 Gb/s for each link
Queuing mechanism	Hierarchical Token Bucket (HTB)

Figure 4 depicts the obtained results in scenario 1 (in ms and μs) regarding γ, and α_{ij} packets type and number. Indeed, figure 4a, 4b and (4c, 4d, 4e) shows the (minimum, average and maximum) E2E latency experienced on scenario 1 by SDNSP, SDQoS and (SDLL, respectively) for HP, MP, and LP (α_{i1} is HP, α_{i2} is MP, and α_{i3} is LP) packets with different γ relative to HP queue load (80%, 50% and 30%) and α_{ij} ($\alpha_{1j} = 5 \times 10^4$, $\alpha_{2j} = 2.5 \times 10^4$,and $\alpha_{3j} = 5 \times 10^3$). It is worth mentioning that SDNSP and SDQoS are traffic type agnostic solutions. We make three main observations here.

- *Low latency is guaranteed by SDLL*: As expected in these figures, the latency is bounded under $649.9\,\mu s$ for HP, MP, and LP traffic. As mentioned above, for SDNSP and SDQoS, packets of all traffic types traverse the shortest path (i.e., from switch 1 to switch 2 to switch 3 as shown in Figure 3) until it gets saturated, so the packets split to other longer paths (such as from switch 1 to switch 4 to switch 5 to switch 3) which increase the latency. The value is less than $34.4\,ms$, and this represents a factor of almost ×53 in comparison with SDLL value. We also notice a reduction in latency on this experience when $\alpha_{i.}$ is getting low which is related to the underload of the system (queues and links). Furthermore, we can see a difference in the latency between each type of traffic when using SDQoS compared to SDNSP, and this is related to the prioritization of HP and MP over LP packet by setting the HP, MP and LP queues maximum rate to (50%, 30%, and 20%) of the total bandwidth respectively.
- *(Almost) similar results were seen for each priority*: We observe that the average latency of the three types of traffic (i.e., HP, MP and LP) is comparable with a small tendency to prioritize the HP packets over the MP packets and the MP over the LP packets. Indeed, exploiting the same high bandwidth offered through the different switches, hence providing several paths, the SDLL could handle incoming packets at each switch faster and without significant queuing delays. It can also be noticed that when the network is loaded, the latency experienced by packets of the same traffic category does not fluctuate hugely ($0.3ms$ with SDLL and $32.16ms$ with SDNSP) and remains stable below a target value, which is in line with the core spirit of deterministic networking.
- *Very low latency is guaranteed by reducing the threshold*: Indeed we can see from the pictures that the latency of HP packets is correlated with the threshold value related to hop-by-hop queue load. Thus, a low threshold could be used for strict priority traffic to make a ceil on the queues load. Hence this will reduce the queuing delay and guarantee very low latency for that type of traffic.

Figure 5 depicts the Cumulative distribution function (CDF) of latency (in ms and μs) obtained for HP and HP$^+$ packets as addressed in scenario 2. It shows as in scenario 1, a bounded latency around $650\,\mu s$ when SDLL is in use,$19\,ms$ for SDQoS and $25\,ms$ for SDNSP with more advantage for HP$^+$ traffic over HP in SDLL related to the threshold value. The results reveal that our solution is well suited for different numbers and types of traffic. Indeed, in SDLL, the entire HP and HP$^+$ traffic are sent over the shortest path, which is from switch 1 to switch 2 to switch 3. When the queues in this path are overloaded, new queues are created on this path, and others



(a) SDNSP

(b) SDNQoS

(c) SDLL

Figure 5: Scenario 2 E2E latency results experienced by HP and HP$^+$ packets when using SDNSP, SDQoS and SDLL with $\gamma = (80\%, 50\%, 30\%)$ for HP packets.

are updated as explained previously in the Algorithm 1.

Figure 6 shows the latency variation during a period of time when using SDLL. We considered a sample of 200 packets sent over 50,000 HP and HP$^+$ packets for each host during a period of 1.3 seconds. It shows the adaptability of the solution regarding the variation of the latency, which increases when the HP queue is filling up, and when γ is exceeded ($time = 0.24s$), a new queue is created with a high maximum rate, while other queues are updated to reduce the average E2E latency of HP$^+$ packets.

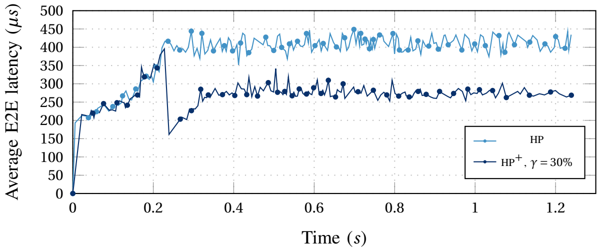


Figure 6: SDLL latency variation during a period of time.

These results can confirm the efficiency of the TE module for guaranteeing a very low E2E latency bound for a different type of traffics. They also show the power and the adaptability of the QM module for the same purpose, which is the main contribution of SDLL.

V. Conclusion

In this paper, we introduced SDLL solution for QoS provisioning and latency guaranty in 5G and beyond. The main purpose was to explore all the E2E paths in the 5G TN and choose the optimal ones in terms of the number of hops, number of queues, and queues load in a dynamic and seamless way. It allows TE and QM in real time to guarantee low latency according to traffic load and priority. Results show that our solution outperforms two reference solutions. Our future focus will be working on a theoretical way to set the best thresholds and implementing this solution on an open-source hardware target (NetFPGA) [13] to demonstrate its efficiency in a real and scaled environment.

Acknowledgment

This work was partially supported by the European Union's Horizon 2020 Research and Innovation Program under the 5G!Drones project (Grant No. 857031).

References

[1] I.-T. N. 2030, "A blueprint of technology, applications and market drivers towards the year 2030 and beyond," 2019. [Online]. Available: https://www.itu.int/en/ITU-T/focusgroups/net2030/Documents/White_Paper.pdf

[2] A. Ksentini, P. A. Frangoudis, A. PC, and N. Nikaein, "Providing low latency guarantees for slicing-ready 5g systems via two-level MAC scheduling," *IEEE Netw.*, vol. 32, no. 6, pp. 116–123, 2018. [Online]. Available: https://doi.org/10.1109/MNET.2018.1800005

[3] K. Boutiba, A. Ksentini, B. Brik, Y. Challal, and A. Balla, "Nrflex: Enforcing network slicing in 5g new radio," *Comput. Commun.*, vol. 181, pp. 284–292, 2022.

[4] L. Zhao, P. Pop, and S. S. Craciunas, "Worst-case latency analysis for IEEE 802.1qbv time sensitive networks using network calculus," *IEEE Access*, vol. 6, pp. 41 803–41 815, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2858767

[5] Z. Zhou, J. Lee, M. S. Berger, S. Park, and Y. Yan, "Simulating TSN traffic scheduling and shaping for future automotive ethernet," *J. Commun. Networks*, vol. 23, no. 1, pp. 53–62, 2021. [Online]. Available: https://doi.org/10.23919/JCN.2021.000001

[6] X. Yin, Y. Liu, L. Yan, and D. Li, "Qos flow mapping method of multi-service 5g communication for urban energy interconnection," in *2021 International Conference on Wireless Communications and Smart Grid (ICWCSG)*, 2021, pp. 75–78.

[7] A. N. Abbou, T. Taleb, and J. Song, "A software-defined queuing framework for qos provisioning in 5g and beyond mobile systems," *IEEE Netw.*, vol. 35, no. 2, pp. 168–173, 2021.

[8] Y. Goto, B. Ng, W. K. G. Seah, and Y. Takahashi, "Queueing analysis of software defined network with realistic openflow-based switch model," *Comput. Networks*, vol. 164, 2019.

[9] R. Kumar, M. Hasan, S. Padhy, K. Evchenko, L. Piramanayagam, S. Mohan, and R. B. Bobba, "End-to-end network delay guarantees for real-time systems using SDN," in *2017 IEEE Real-Time Systems Symposium, RTSS 2017, Paris, France, December 5-8, 2017.* IEEE Computer Society, 2017, pp. 231–242.

[10] J. Prados-Garzon, T. Taleb, and M. Bagaa, "LEARNET: reinforcement learning based flow scheduling for asynchronous deterministic networks," in *2020 IEEE International Conference on Communications, ICC 2020, Dublin, Ireland, June 7-11, 2020.* IEEE, 2020, pp. 1–6.

[11] 3GPP TS 23.501, "System architecture for the 5g system, v16.1.0," 2019.

[12] J. Henry, T. Szigeti, and L. M. Contreras, "Diffserv to QCI Mapping," Internet Engineering Task Force, Internet-Draft draft-henry-tsvwg-diffserv-to-qci-04, Apr. 2020, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-henry-tsvwg-diffserv-to-qci/04/

[13] N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore, "Netfpga sume: Toward 100 gbps as research commodity," *IEEE Micro*, vol. 34, pp. 32–41, 09 2014.