SDN-based L4S Congestion Control in Beyond 5G

Sofiane Messaoudi*, Adlen Ksentini*, Franck Messaoudi†, Christian Bonnet*
*Eurecom Institute, †OpenAirInterface Alliance
* †Sophia Antipolis, France
Email: *name.surname@eurecom.fr, †name.surname@openairinterface.org

***Abstract*—This paper describes the SDN-based L4S solution, a congestion control algorithm designed to improve QoS in 5G and Beyond networks. Inspired by the IETF specifications, our framework tackles challenges prevalent in immersive applications like video streaming and cloud gaming, such as ultra-low latency and packet loss. The proposed solution seamlessly integrates L4S techniques into SDN, thereby optimizing queue management within the transport network. Additionally, it employs Explicit Congestion Notification (ECN) to mark packets during congestion scenarios. This synergistic approach facilitates dynamic adjustments in transmission rates, enhancing the overall efficiency of the transport network, particularly in accommodating various classes of traffic. Evaluation results demonstrate that our solution outperforms the benchmarks by a substantial margin in terms of End-to-End latency, and packet loss.**

I. Introduction

The emerging 5G and Beyond (5G&B) networks have tailored the market with new services, including Virtual Reality (VR), Augmented Reality (AR), and diverse forms of video streaming (ranging from gaming to conferencing and entertainment). These real-time applications necessitate hard requirements for latency, packet loss, and throughput to ensure seamless, high-volume data streaming for a fully immersive user experience [1]. Transported over Transmission Control Protocol (TCP), the protocol's congestion control mechanism dynamically adjusts the TCP window size to regulate data transmission rates, impacting throughput. Indeed, upon detecting network congestion, often signaled by packet loss, the TCP congestion control algorithm iteratively reduces its window size until congestion recovery. Besides, TCP employs the *slow start* mechanism to gradually increase transmission rates, which may take time to reach optimal throughput. TCP exhibits notable limitations, including aggressive rate reduction leading to under-utilization of bandwidth and sub-optimal throughput, delayed congestion detection resulting in packet loss and increasing End-To-End (E2E) latency, and resource allocation inequity between different TCP streams [2].

To address these limitations, the Internet Engineering Task Force (IETF) has introduced a new architecture, currently under specification, known as Low-Latency, Low-Loss, Scalable Throughput (L4S) [3]. This approach leverages the Explicit Congestion Notification (ECN) to mark packets [4] and uses a dual-queue framework within switches to distinguish and prioritize L4S traffic. By using the ECN and dual-queue logic, the L4S architecture reduces packet loss, increases throughput, and decreases E2E latency. Indeed, for the packet loss problem, the L4S architecture avoids packet dropping and re-transmission by predicting and preventing congestion (i.e., packet marking). For the high throughput, it adjusts the transmission rate at the Sender derived from congestion probability estimates provided by the Receiver. Besides that, the use of an additional queue sorts the traffic within the two queues (regular and L4S), which reduces the number of packets per queue and leads to reduced queuing delay hence the E2E latency.

E2E latency in packet-switched networks is mostly the *sum of 4 delays* known as processing (t_{proc}), queuing (t_{queue}), transmission (t_{trans}), and propagation (t_{prop}) delays [2]. t_{proc} is the required time to parse packets, verify their checksum, and direct them. t_{queue} is the time a packet spends in a queue before being transmitted onto the link; it is impacted by the queue size and the number of packets within the queue. t_{trans} represents the needed time to push all the packet's bits into the link; determined by packet size and transmission rate ratio. Finally, t_{prop} is the time needed to cross a link between two nodes. t_{prop} is obtained as a ratio of distance and link speed. While t_{proc}, t_{trans}, and t_{prop} are relatively negligible (order of μs to few ms) compared to t_{queue}, most of efforts to reduce the E2E latency have focused on various solutions including, hardware acceleration [5], parallel processing [6], and computation offloading [7], for t_{proc}. Solutions such as Gigabit Ethernet cards ($\times 10$, $\times 100$ $Gbps$), and (extended) Berkeley Packet Filtering ((e)BPF) eXpress Data Path (XDP) [8] address t_{trans}, whilst t_{prop} is reduced via optical fiber with repeaters and amplifiers, Multi-Access Edge Computing (MEC) [9], Multi-Path TCP (MP-TCP) [10], and path optimization algorithms. Improvements to t_{queue} have been made through Quality of Service (QoS), Queuing Discipline (qdisc), and Traffic Control (tc) implementations [11], and ECN L4S application.

Several studies have explored the potential of L4S and ECN in congestion control. In [12], the authors conducted a comprehensive study on L4S, introducing a congestion control scheduler aimed at enhancing latency and throughput without impacting classical traffic. Despite its innovative features, this solution has yet to be deployed in real-world scenarios. N. Nguyen et al. [13] employed P4 and in-band network telemetry to monitor L4S switch metrics. Their experimentation yielded promising results, showing minimal processing overhead and interesting results, albeit with limitations in deployment flexibility due to reliance on P4 switches. In a separate study, authors in [14] addressed challenges arising from heavy traffic and impairments in data centers, such as TCP-Incast, buffer overflow, and long queues, and proposed an enhanced algorithm leveraging ECN. While effective within

data center environments, the applicability of this approach to broader network contexts is limited. Furthermore, [15] introduced a congestion control algorithm aligned with L4S specification, implemented in Web Real-Time Communication (WebRTC), which used ECN for adaptive sending rate adjustments. Comparative evaluations against Google Congestion Control (GCC) baseline demonstrated improved responsiveness. Nevertheless, its notable dependency on manual adjustments and hardware/kernel modifications posed scalability challenges.

Despite the considerable efforts made in these studies, a common limitation to our best knowledge appears to persist. None of these works have fully considered a holistic perspective of the network, which is crucial to accurately identifying bottlenecks. Surprisingly, Software-Defined Networking (SDN) paradigm was not applied through these works with regard to its significance in orchestrating control across the entire network. In this paper, we propose to integrate L4S techniques within an *´sdnized'* network. We enhanced the efficiency of Queue Management (QM) and ECN marking. This integration allows the server to adjust the sending rate dynamically, guaranteeing ultra-low latency and elevated QoS levels, especially for time-sensitive applications. Overall, this paper makes two contributions. Firstly, it presents a novel SDN-L4S framework, embedding a new approach to low-latency networking that capitalizes on the strengths of SDN, tailored specifically for the demands of 5G&B in scenarios marked by high network loads. Secondly, it uses an adaptive QM addressing the intricate requirements of diverse delay-critical services; we introduce a dynamic QM algorithm that enforces priority, safeguarding the L4S traffic even with the presence of normal traffic.

The rest of the paper is organized as follows: Section II introduces the background. Our solution is shown in Section III and evaluated in Section IV. Section V concludes the paper.

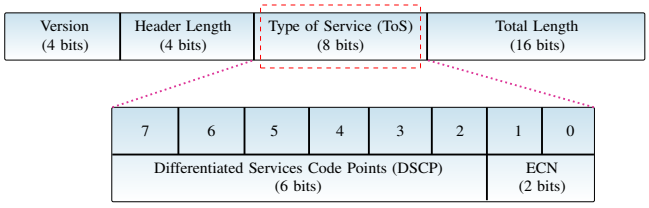
II. Background

In TCP/IP networks, congestion is signaled by packet drops, an effective yet performance-degrading mechanism. Thus, we need alternative methods to communicate congestion information to endpoints. In this section, we introduce the concepts of ECN and L4S, which form the core of our solution.

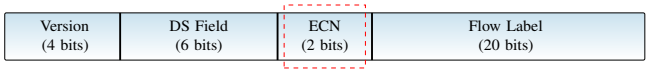
A. ECN

ECN serves as a mechanism to signal or anticipate network congestion. It facilitates E2E congestion notification between two endpoints (i.e., Sender and Receiver) within TCP/IP-based networks. For optimal E2E congestion control, all the nodes involved in the transmission path including the endpoints *must* be ECN-enabled. The presence of any device along the path that does not support ECN breaks down the E2E ECN functionality.

The ECN congestion notifications aim to reduce packet loss and delay by prompting the Sender to reduce transmission rates without dropping packets. The IETF has introduced the document RFC 3168 [4], which describes how the ECN is added to the Internet Protocol (IP) header. IETF has reserved two bits in the IP header's Type of Service (ToS) field or the Differentiated Services Code Point (DSCP) field (see Figure 1) to signal congestion more explicitly and proactively. The nodes within the network (routers, switches, access points, and base stations) set the ECN field to 11 (that is, Congestion Experienced (CE)) when detecting increased congestion, providing advanced notice to the Sender and fostering congestion feedback with minimal packet loss. The Sender interprets the bits of the ECN and adjusts its transmission rates or congestion control algorithms accordingly.



(a) IPv4 Header, First 4 Bytes Focusing on ToS



(b) IPv6 Header, First 4 Bytes

Figure 1: ECN Bits in the IP Headers

B. L4S

L4S is an innovative technology intended to guarantee high throughput while minimizing delay and packet loss in Internet traffic. It is based on the idea of signaling early on congestion (CE) when the number of queued packets in a network node surpasses a predefined threshold. This approach ensures a consistently low queue delay, reducing the E2E latency while optimizing link utilization. This technology relies mainly on two main mechanisms, namely *congestion control* and *transport feedback*.

ECN is the central element in the *congestion control* mechanism. Table I shows the possible values of ECN and their meaning. A setting of 00 or 10 in the ECN field indicates that one or more network nodes in the path do not support L4S capability. On the contrary, the values 01 and 11 denote the presence of L4S traffic, with all nodes of the network along the path supporting L4S capability, at the difference that 11 is used to express congestion on the path.

Table I: L4S Codepoints and Meaning

Binary Codepoint	Codepoint Name	Description
00	Not-ECT	Not ECN-capable transport
01	ECT(1)	L4S-capable transport
10	ECT(0)	Not L4S-capable transport
11	CE	Congestion Experienced

Transport feedback represents the transport protocols used to alert the Sender accurately about congestion. Today's most transport protocols support the ECN feedback, we can cite TCP [16], Stream Control Transmission Protocol (SCTP) [17], Quick UDP Internet Connections (QUIC) [18], and Real-time Transport Protocol (RTP) [19]. Unlike traditional ECN, L4S

promptly marks packets and smoothens congestion feedback for each marked packet.

Figure 2 illustrates the key components of the L4S architecture and their interactions. A server, embodying the L4S Sender, responds to a client by generating streams with a dynamically determined transmission rate $\lambda(t)$ in each round trip. The client serves as the L4S Receiver, consuming the Sender's streams and subsequently providing congestion feedback. Furthermore, the illustration emphasizes the importance of having L4S-capable transport nodes along the path between the L4S Sender and Receiver.

In congestion signaling and rate adaptation, the process unfolds as follows: The Sender signals L4S support using ECN codepoint *ECT(1)*. Network nodes then recognize the packet as an L4S packet and, in case of congestion, alter the ECN bits to indicate *CE*. Upon reaching the Receiver, if the ECN bits indicate congestion along the transmission path, the Receiver notifies the Sender. In response to the congestion notification, the Sender reduces the transmission rate.

In protocols like TCP, transmission rate reduction occurs through congestion control mechanisms, such as TCP Friendly Rate Control (TFRC) [20], Additive Increase/Multiplicative Decrease (AIMD) [21] scheme, along with other schemes including slow start [22] and Congestion Window (CWND) [23]. These algorithms adjust the Sender's transmission rate based on received ACKnowledgments (ACKs) and congestion indicators, including ECN markings.

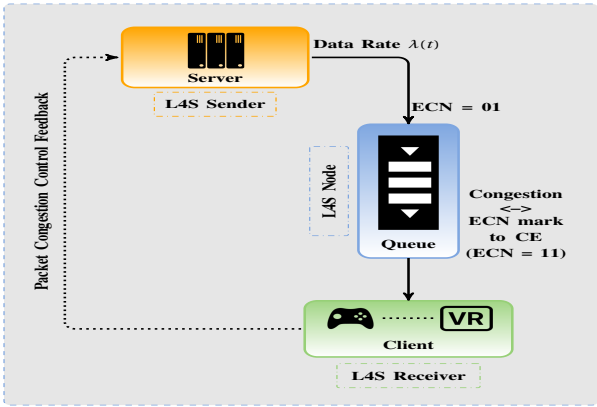


Figure 2: L4S Mechanism

III. SDN-L4S Solution

In this section, we provide an overview of our proposal, detailing its design and workflow.

A. Solution Overview

The solution is designed within a distributed architecture involving an L4S Sender, an L4S Receiver, and an SDN controller. It comprises the components shown in Figure 3. In the following we describe the components of interest. Note that our solution is applied to 5G&B. For more details on the SDN architecture and its applicability to 5G&B, we encourage readers to refer to our previous work [24].

- *Queue Manager:* It configures an *on-demand* L4S and/or classical queue with a $queue_{id}$ based on traffic type. L4S traffic is directed to L4S queue and regular traffic to classical queue. Each switch port may have at most two queues, with the L4S queue enjoying superior priority and output rate.
- *ECN Marker:* It applies ECN markings to packets based on congestion conditions.
- *Flux Rules Manager & Path Selector:* It manages and processes fl ux rules, determining the optimal path and queue for traffic based on predefined policies and network conditions.
- *Packet Interceptor:* It intercepts incoming packets, allowing for real-time analysis.
- *Packet Parser:* Manipulates packet headers and payload contents, and read field of interest.
- *Rate Calculator:* It dynamically regulates the packet transmission rate $\lambda(t)$. This controls the data flow, preventing rapid bursts and contributing to smoother and more efficient data transfer.
- *Ingress/Egress Listener:* Sends/receives L4S/regular packets and receives congestion feedback's from the L4S Receiver.
- *Congestion Probability Calculator:* Upon getting L4S packets, the L4S Receiver calculates the probability of congestion and notifies the Feedback Agent.
- *Feedback Agent Sender/Receiver:* It sends/receives feedback information to/from the L4S Sender/Receiver via *In-Band Channel* by extending the TCP header and setting the optional field to express the congestion probability.

B. Notation

In this section, we present a mathematical notation outlining our solution. We define an OpenFlow rule, denoted as ξ (see Equation 1), as a composite of essential components [25]:

(i) The *Match set* (μ) (see Equation 2), which characterizes the packet's ToS value. Used to classify flows into two categories: *L4S* traffic, identified by $\mu = 185$, and *regular* (Non-L4S) traffic, where $\mu \neq 185$.

(ii) The *Action set* (α) (see Equation 3) specifies the action to be executed for each flow packet. We identify three main actions. The first action directs L4S traffic to the L4S queue, while regular traffic goes throw the classical queue based on the $queue_{id}$ parameter. The second action is used in the event of queue congestion detection (in L4S or classical queue) to mark each packet with the ECN value equal to 11. Congestion is determined when the queue load (γ) exceeds a predefined threshold (q_1) of the queue size (σ). The third action differentiates between congestion and non-congestion events; only regular traffic is subject to dropping in case of congestion; elsewhere, the traffic, whether L4S or not, is accepted and forwarded to the Receiver through a specified set of port numbers.

(iii) *Priority* (δ) (Equation 4) that is used to order rules in the forwarding switch.

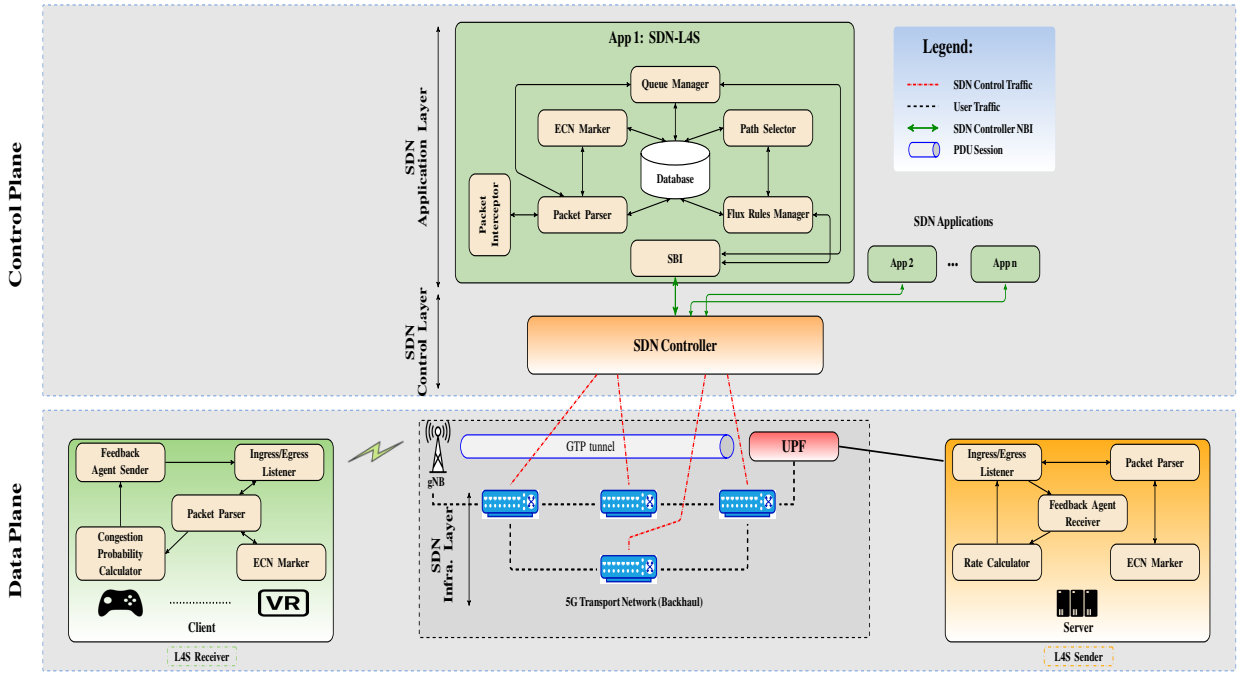


Figure 3: SDN-L4S Solution applied to 5G&B

$$\xi = \{ [\mu], [\alpha], \delta \} \quad (1)$$

$$\mu = \{0, 1, 2, ..., 224\} \quad (2)$$

$$\alpha = \{ setECN(value),\ output(port_{n^\circ}, queue_{id}),\ drop \} \quad (3)$$

$$\delta = \{0, 1, 2, ..., 65535\} \quad (4)$$

Equation 5 defines the new rate of packets departure λ dynamically adjusted by the L4S Sender in response to congestion. This new rate, measured in $Mbps$, is intricately linked to the probability of congestion, denoted as p (see Equation 6, in which n is the number of received packets in congestion state (i.e., packets marked with ECN = 11) and N is the total number of received L4S packets (i.e., L4S traffic with $\mu = 185$)). The probability p is computed on the L4S Receiver side for each new packet arrival and used to guide the L4S Sender's rate adaptability using a predefined threshold (q_2). Indeed, in each system, the probability of system occupancy $p \approx \frac{\lambda_a}{\lambda_a + \lambda_d}$, where λ_a is the arrival rate to the system and λ_d is the departure rate [26]. This can be further simplified as $p \approx \frac{1}{1+\lambda_d}$, ultimately leading to the equation 5.

$$\lambda \approx \begin{cases} \lambda_0 & \text{if } 0 \leqslant p < q_2 \\ \frac{1}{p} - 1 & \text{if } q_2 \leqslant p < 1 \end{cases} \quad (5)$$

$$p = \begin{cases} 0 & \text{if } N = 0 \\ \frac{n}{N} & \text{if } N \neq 0 \end{cases} \quad (6)$$

C. Workflow

The proposed SDN-L4S algorithm aims to prevent congestion in an SDN-managed network by integrating adaptive rate control, QM, and congestion feedback mechanisms. The algorithm takes into account the following inputs: the network topology information (encompassing paths {Paths}, hosts {Hosts}, and link characteristics {Links}), the set of available flux rules {ξ}, the real-time queue statistics {Stats}, the initial transmission rate λ_0 matching the speed of Network Interface Controller (NIC), and the congestion-related counters (n, N, p) initialized to zero. Outputs from the algorithm include updated flux rules, a probability of congestion (p), and a revised transmission rate λ. The algorithm is partitioned into the following sections.

L4S Sender: The logic behind this component is to dynamically fine-tune responses to the Receiver according to a designated rate λ. This rate undergoes *near real-time* adjustments, where, with each newly received congestion probability (p), the Sender recalibrates the transmission rate (λ) by applying Equation 5 (see Algorithm 1).

Algorithm 1 SDN-L4S Sender Algorithm

1: **Inputs:** λ_0, $p_0 = 0$, p, q_2
2: **Outputs:** λ
3: if ($p \geqslant q_2$) **then**:
4: Calculate the new rate: $\lambda = \frac{1}{p} - 1$
5: else
6: Re-initialize the new rate: $\lambda = \lambda_0$
7: Adjust transmission rate to λ
8: Send next packet

SDN Controller: Performs actions on traffic and manages switch queues based on processed packet information (μ) received from PACKET-IN messages. If no suitable path adhering to the specified configuration ξ for the given μ, alternative routes are explored. If no path is found, the destination is

labeled as unreachable. Subsequently, the controller manages both L4S and regular traffic. For L4S traffic, it handles the L4S queue, creating one if needed, and monitors its load. If the load exceeds a predefined threshold (q_1), the packet is marked as congested (ECN = 11) and forwarded. Meanwhile, the regular traffic is routed through the standard queue, with packets marked as congested and dropped if the queue load exceeds q_1. A new configuration ξ is applied to the switches (see Algorithm 2).

Algorithm 2 SDN-L4S Controller Algorithm

```
Inputs: {Paths}, {Hosts}, {Links}, {ξ}, {Stats}
Outputs: {ξ}
SDN CTRL receives PACKET-IN message with μ info
if (exists path with a ξ satisfying packet μ) then:
    Apply ξ
else
    Find all potential paths to the destination
    if path list is empty then:
        The destination is unreachable
    else
        if (μ = 185) then:
            Create L4S queue if not exists
            if (load (L4S queue) ⩾ q1) then:
                Mark packet as congested: ECN = 11
        else
            Choose the path with a regular queue
            if (load(regular queue) ⩾ q1) then:
                Mark packet as congested: ECN = 11
                Drop the packet
        Install the ξ on switches
```

L4S Receiver: The main action of this component is to prevent congestion and warn the L4S Sender to reduce its transmission rate. A probability of congestion p for L4S traffic is determined by the ratio of n, representing the received packets flagged as congested (ECN = 11), to the total number of L4S packets N or simply set to zero (using equation 6). The counters N and n are then updated accordingly (see Algorithm 3).

Algorithm 3 SDN-L4S Receiver Algorithm

```
Inputs: n = 0, N = 0
Outputs: p
Handle received packet:
if (μ = 185) then:
    N++
    if (ECN = 11) then:
        n++
    Calculate the congestion probability: p = n/N
Notify the L4S Sender of the updated value of p
```

IV. Performance Evaluation

A. Setup

Our framework has been tested within a linear topology with 2 Open vSwitch (OVS) switches (v2.13.5). We have used 24 Linux nodes (Mininet v2.2.2) as traffic source generators, where 12 represent L4S Senders and the other 12 for regular Senders. Each Sender generates a TCP stream of size 65,507 Bytes. We have distinguished the entire traffic within 2 sets according to ToS (μ) (12 flows with $\mu = 185$ and 12 flows with $\mu \neq 185$). We also used a Linux node as an L4S Receiver that receives all the generated traffic in a competition mode. As SDN controller, we used ONOS (v2.7.0) with OpenFlow (v1.6). The tests have been repeated 100 times with a number of packets generated of 24×10^4 packet for each iteration (10^4 packet for each of the 24 hosts). For the bandwidth of the links, it was fixed to $10\,Gbps$.

B. Results

Now, we present the results of the measurement campaign regarding the E2E latency and Packet Loss Rate (PLR) impacted by queue load, congestion probability, as well as QM.

1) E2E Network Latency: In this section, we analyze the causes that affect the latency.

Congestion Probability Threshold Impact: Figure 4 shows the impact of congestion probability threshold q_2 on the average E2E latency (in ms) for both SDN-L4S and SDN-Traditional Congestion Control (TCC), with the queue load threshold q_1 held constant at 80%. We may notice that for all q_2, SDN-L4S latency is *always* shorter than SDN-TCC latency, attributed to its adaptive transmission rate that reduces the queuing delay. Furthermore, as q_2 increases, the SDN-L4S latency gradually rises, eventually approaching the SDN-TCC latency as q_2 approaches 100%. Conversely, SDN-TCC demonstrates higher and relatively stable latency, attributed to its packet drop mechanism when q_1 is reached, regardless of the value of q_2.

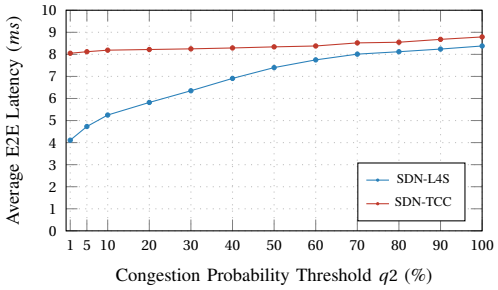


Figure 4: E2E Average Latency Versus q_2 ($q_1 = 80\%$)

Queue Load Threshold Impact: Figure 5 shows the impact of queue load threshold q_1 on the average E2E latency (in ms) for both SDN-L4S and SDN-TCC, with the congestion probability threshold q_2 fixed at 1%. We remark that SDN-L4S latency is shorter than SDN-TCC once, and this is true for all q_1 values, thanks to the L4S mechanism. Besides that, SDN-L4S latency is approximately $\times 1/2$ that of SDN-TCC. Despite

the fact that q_1 impacts both solutions, SDN-L4S responds better to congestion because it is based on the early signaling of congestion (q_2).

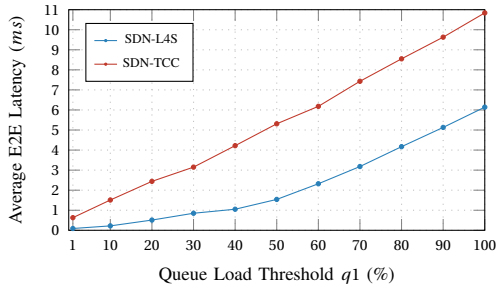


Figure 5: E2E Latency Versus q_1 ($q_2 = 1\%$)

Queue Management Impact: Figure 6 highlights the impact of Queue Management (QM) on the E2E latency. It displays the min, max, and median values (in *ms*) obtained for both SDN-L4S and SDN-TCC with/without QM (i.e., SDN-L4S$_{\text{QM}}$, SDN-TCC$_{\text{QM}}$, ¬SDN-L4S$_{\text{QM}}$, and ¬SDN-TCC$_{\text{QM}}$, respectively). We have fixed in this scenario the couple q_1 and q_2 at 80% and 1%, respectively. Initially, latency values are consistently lower across all scenarios during the early phase of traffic transmission before queues become congested. It is worth noting that when QM is employed, a dedicated queue is created for L4S traffic with higher priority and transmission rate, resulting in notably lower latency for SDN-L4S$_{\text{QM}}$ than other scenarios. Conversely, ¬SDN-L4S$_{\text{QM}}$ presents better latency then SDN-TCC, thanks to the ECN mechanism. Furthermore, SDN-TCC$_{\text{QM}}$ demonstrates smaller values compared to ¬SDN-TCC$_{\text{QM}}$, as it avoids congestion caused by all traffic (i.e., L4S and regular) competing for the same queue.

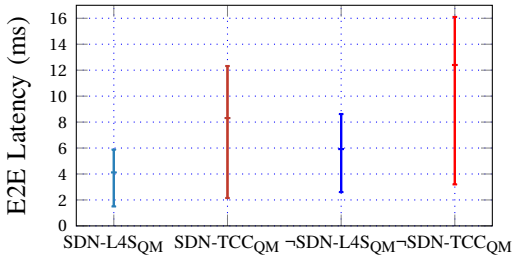


Figure 6: QM Impact on the E2E Latency ($q_1, q_2 = 80\%, 1\%$)

2) Packet Loss Rate: Now, we focus on the causes of PLR variability. Table II presents the PLR obtained for SDN-L4S and SDN-TCC under different combinations of q_1 and q_2 (i.e., SDN-L4S$_{(q_1,\, q_2)}$, SDN-TCC$_{(q_1,\, q_2)}$), while $q_1 \in \{50\%, 80\%\}$ and $q_2 \in \{1\%, 10\%\}$. Please notice that SDN-TCC is not impacted by q_2 according to Section IV-B1, so the PLR. The PLR results demonstrate that SDN-L4S surpasses SDN-TCC and this is true for all q_1 and q_2. When q_1 if fixed, SDN-L4S$_{(x,\, 1)}$ PLR is smaller than SDN-L4S$_{(x,\, 10)}$ as the congestion is detected earlier when $q_2 = 1\%$, so the L4S Sender reduces the transmission rate earlier. Whilst, by fixing q_2, SDN-L4S$_{(50,\, x)}$ PLR is smaller than SDN-L4S$_{(80,\, x)}$, this is obvious since the congestion probability increases slower when q_1 is high (please refer to Algorithm 2 lines 13-14 and Algorithm 3 lines 6-8). Lastly, SDN-TCC$_{(50,\, x)}$ PLR is higher than SDN-TCC$_{(80,\, x)}$ due to drop mechanism that occurs frequently for $q_1 = 50\%$ (Algorithm 2 lines 17-19).

Table II: Packet Loss Rate (in %)

$SDN-L4S_{(q_1,q_2)}$	$PLR(\%)$	$SDN-TCC_{(q_1,q_2)}$	$PLR(\%)$
$SDN-L4S_{(50,1)}$	0.16	$SDN-TCC_{(50,x)}$	24.2
$SDN-L4S_{(50,10)}$	0.5		
$SDN-L4S_{(80,1)}$	2.66	$SDN-TCC_{(80,x)}$	18.4
$SDN-L4S_{(80,10)}$	3.7		

V. Conclusion

In this paper, we have designed a congestion control solution inspired by the IETF standards on the L4S, that we applied to *SDNized* network tailored for video streaming applications in Beyond 5G (B5G). We have demonstrated through the measurement campaign that our solution outperform TCP congestion control mechanisms in term of latency and packet loss. As future work, we aim to explore how to integrate the L4S mechanism within the Radio Access Network (RAN). Besides that, we would like to investigate resource usage and identify a trade-off between queue loads and congestion probability.

Acknowledgment

This work was partially supported by the European Union's Horizon Europe Research and Innovation programme AC^3 project under grant agreement No 101093129.

References

[1] F. Messaoudi, A. Ksentini, G. Simon, and P. Bertin, "Performance analysis of game engines on mobile and fixed devices," *ACM Trans. Multim. Comput. Commun. Appl.*, vol. 13, no. 4, pp. 57:1–57:28, 2017.

[2] J. F. Kurose and K. W. Ross, *Computer networking - a top-down approach featuring the internet*. Addison-Wesley-Longman, 2001.

[3] M. Bagnulo and G. White, "Low latency, low loss, and scalable throughput (l4s) internet service: Architecture," 2023.

[4] S. Floyd, D. K. K. Ramakrishnan, and D. L. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168, Sep. 2001.

[5] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: programming protocol-independent packet processors," *Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.

[6] P. A. Levis, "Tinyos: An open operating system for wireless sensor networks (invited seminar)," in *7th International Conference on Mobile Data Management (MDM 2006), Nara, Japan, May 9-13, 2006*. IEEE Computer Society, 2006, p. 63.

[7] F. Messaoudi, A. Ksentini, and P. Bertin, "Toward a mobile gaming based-computation offloading," in *2018 IEEE International Conference on Communications, ICC 2018, Kansas City, MO, USA, May 20-24, 2018*. IEEE, 2018, pp. 1–7.

[8] A. Deepak, R. Huang, and P. Mehra, "ebpf/xdp based firewall and packet filtering," in *Proc. Linux Plumbers Conf.*, 2018, pp. 1–5.

[9] ETSI, "Mobile edge computing (mec); framework and reference architecture," *ETSI GS MEC*, vol. 3, p. V1.1.1, March 2016.

[10] A. Abdelsalam, M. Luglio, C. Roseti, and F. Zampognaro, "Linux mptcp performance evaluation in a combined terrestrial-satellite access," in *2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*, 2019.

[11] R. Rosen, *Linux kernel networking: Implementation and theory*. Apress, 2014.

[12] K. De Schepper, M. Bagnulo, and G. White, "Rfc 9330: Low latency, low loss, and scalable throughput (l4s) internet service: Architecture," 2023.
[13] H. N. Nguyen, B. Mathieu, M. Letourneau, G. Doyen, S. Tuffin, and E. M. d. Oca, "A comprehensive p4-based monitoring framework for l4s leveraging in-band network telemetry," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, 2023, pp. 1–6.
[14] E. Gilliard, K. Sharif, A. Raza, and M. M. Karim, "Explicit congestion notification-based congestion control algorithm for high-performing data centers," in *2023 IEEE AFRICON*, 2023, pp. 1–6.
[15] J. Son, Y. Sanchez, C. Hampe, D. Schnieders, T. Schierl, and C. Hellge, "L4s congestion control algorithm for interactive low latency applications over 5g," in *2023 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2023, pp. 1002–1007.
[16] M. Kühlewind, R. Scheffenegger, and B. Briscoe, "Problem Statement and Requirements for Increased Accuracy in Explicit Congestion Notification (ECN) Feedback," RFC 7560, Aug. 2015.
[17] R. R. Stewart, "Stream Control Transmission Protocol," RFC 4960, Sep. 2007.
[18] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021.
[19] "RTP Control Protocol (RTCP) Feedback for Congestion Control," RFC 8888, Jan. 2021.
[20] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "Tcp friendly rate control (tfrc): Protocol specification," Tech. Rep., 2003.
[21] P. Hurley, J.-Y. Le Boudec, and P. Thiran, "A note on the fairness of additive increase and multiplicative decrease," Tech. Rep., 1998.
[22] H. Wang, H. Xin, D. S. Reeves, and K. G. Shin, "A simple refinement of slow-start of tcp congestion control," in *Proceedings ISCC 2000. Fifth IEEE Symposium on Computers and Communications*. IEEE, 2000, pp. 98–105.
[23] H. Torkey, G. Attiya, and A. A. Nabi, "An efficient congestion control protocol for wired/wireless networks," *International Journal of Electronics Communication and Computer Engineering*, vol. 5, no. 1, p. 77, 2014.
[24] S. Messaoudi, A. Ksentini, F. Messaoudi, and C. Bonnet, "Gnn-based sdn admission control in beyond 5g networks," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE, 2023, pp. 6103–6108.
[25] 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)*. IEEE, 2023, pp. 587–592.
[26] K. De Schepper and G. White, "Rfc 9332: Dual-queue coupled active queue management (aqm) for low latency, low loss, and scalable throughput (l4s)," 2023.