2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring) | 978-1-7281-8964-2/20/$31.00 ©2021 IEEE | DOI: 10.1109/VTC2021-Spring51267.2021.9448764

Low Latency Low Loss Scalable Throughput in 5G Networks

Davide Brunello*, Ingemar Johansson S†, Mustafa Ozger*, and Cicek Cavdar*
*School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden
E-mail: {brunello, ozger, cavdar}@kth.se
†Ericsson AB, Luleå, Sweden
E-mail: ingemar.s.johansson@ericsson.com

***Abstract*—Low Latency Low Loss Scalable Throughput (L4S) is a technology intended to reduce queue delay problems, ensuring low latency to Internet Protocol flows with a high throughput performance. To reach this goal, it relies on Explicit Congestion Notification (ECN), a mechanism that marks packets to signal congestion in the network avoiding packets to be dropped. The congestion signals are managed at the sender and receiver sides thanks to scalable congestion control algorithms. In this paper, the challenges to implement L4S in a 5G network are analyzed. Using a proprietary state-of-the-art network simulator, the L4S marking strategy has been implemented at the Packed Data Convergence Protocol layer. To evaluate the benefits of the implementation, L4S has been adopted to support Augmented Reality (AR) video gaming traffic while using the IETF experimental standard Self-Clocked Rate Adaptation for Multimedia (SCReAM) for the congestion control. The results show that the video gaming traffic experiences lower delay when supported by L4S. Moreover, in all the cases analyzed, L4S provides an average application layer throughput above the minimum requirements of a high-rate latency-critical application, even at high system loads. Furthermore, the packet loss rate has been significantly reduced thanks to L4S. If it is used in a combination with a Delay Based Scheduler (DBS), a packet loss rate very close to zero has been reached.**

***Index Terms*—L4S, 5G, low latency, SCReAM, high-rate latency-critical applications.**

I. Introduction

Nowadays, interactive high-rate latency-critical applications such as online gaming, Virtual Reality (VR), Augmented Reality (AR) and remote control are becoming more and more widespread. They require low latency and high throughput to bring a better user experience. Although achieving high throughput is possible, it is challenging to guarantee high throughput and low latency simultaneously.

The latency can be adversely affected by events such as congestion and packet loss. These events occur when more traffic is injected into the network than the current capacity. To avoid these unwanted events, congestion control algorithms have been utilized at the sender and receiver sides (also called endpoints). Active Queue Management (AQM) techniques have been proposed for managing buffers at the network nodes such as routers. To reduce the number of retransmissions and to reach a higher throughput, the network nodes implement relatively large buffers, where packets are queued instead of being dropped. However, the use of large buffers introduces another problem, known as queue delay. It occurs when the input rate of packets is higher than the output rate in a network node. Hence, the incoming packets are queued in the buffer waiting to be processed and sent.

Low Latency Low Loss Scalable Throughput (L4S) is a technology intended to mitigate the queue delay problem, ensuring low latency to Internet Protocol (IP) flows without affecting the throughput performance. The working principle of L4S is to change the Explicit Congestion Notification (ECN) bits in the IP header as soon as the queues in the network nodes start to grow. The action of changing the ECN bits to signal congestion is called *marking*. Hence, it is possible to signal congestion early and to reduce the queue delay. Currently, L4S is in the standardization process at IETF [1].

L4S is a general technology that can be implemented in different types of networks, thus poses a different set of obstacles while implementing it in various networks. In this paper, the implementation challenges of L4S in a 5G network such as changing the ECN bit in the Radio Link Control (RLC) layer and the coexistence of L4S with IP-in-IP tunnels have been analyzed. Afterward, we propose a marking strategy for the congestion notification. L4S with the proposed marking strategy has been implemented in a 5G context with a proprietary network simulator. Finally, the performance improvement with L4S support and the proposed marking strategy has been investigated to support high-rate latency-critical applications such as video applications for AR, which coexist with web traffic. The research questions that will be answered in this paper are the following:

- The first question concerns tunnels used in a cellular network such as General Packet Radio Service (GPRS) Tunnelling Protocol for user plane (GTP-U) and Internet Protocol Security (IPSec) tunnels. Tunnels involve the creation of new packets with new headers, and L4S uses header bits to signal congestion. Do tunnels create any obstruction for the L4S congestion signals propagation?
- Usually in New Radio (NR) packets are queued at the RLC layer. Packets at the RLC layer are encrypted, thus it is impossible to mark the packets to signal congestion. Is there any solution to overcome this problem?
- How to handle classic and L4S traffic simultaneously, is it necessary to use different schedulers with different priorities?
- Given that L4S marks packets early when the queue starts

to grow (based on a threshold), what is the impact of changing the marking thresholds?

- Can L4S improve key performance indicators (KPIs) of high-rate latency-critical application traffic in a 5G context?

The novelty of this paper consists in the implementation and assessment of L4S in a 5G cellular network for the first time in the literature to the best of our knowledge. The contribution regards addressing the implementation challenges, studying the trade-offs between throughput and latency, and finally assessing the performance regarding L4S technology in a 5G network context.

II. Preliminaries for L4S

L4S enables AQM so that network nodes exploit the use of ECN bits in the IP header to notify that congestion has been experienced for the packet routed from a sender to its corresponding receiver. Thanks to a scalable congestion control utilized in the end hosts, it is able to provide a good throughput performance. The following subsections provide detailed information about concepts utilized in the L4S.

A. Explicit Congestion Notification

ECN is a standardized mechanism [2] for the management and notification of congestion in the network, which avoids dropping the packets in the buffer. This mechanism is intended to reduce end-to-end latency by giving a faster and explicit signal of congestion. The operating principle is simple and based on the use of two bits in the IP header, namely ECN bits, to notify congestion in the network. Through these 2 bits, it is possible to encode 4 different codepoints, which are explained in Table I. If a packet is set as the codepoint of Not-ECT by the sender node, then the packet does not support ECN and cannot be marked as congestion experienced (CE) in case a congestion event occurs. Conversely, if the packet has codepoints ECT(0) or ECT(1), this means that the packet supports ECN by the sender node, the network nodes can change the ECN bits to CE to signal congestion. The main requirement for ECN to work correctly is that the network nodes must be able to recognize the traffic that supports ECN, and also be able to modify the ECN bits to signal the congestion.

TABLE I: ECN codepoints and meaning.

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

B. Active Queue Management

AQM is a policy for managing buffers in various nodes in a network, such as routers and switches. It consists in the drop of packets from the queue before approaching its memory limit. The purpose of AQM is to reduce and prevent excessive network congestion and uncontrolled overflow of network buffers or high Round Trip Time (RTT), thus AQM improves performance for end-to-end flows. If the traffic is ECN-capable, AQM can mark packets instead of dropping them. This improves the performance further since packet loss and consequent packet retransmission are avoided. As explained in [3], it is recommended to implement the ECN technique in each node of the network, where the AQM support is implemented.

C. Scalable Congestion Control

Scalable congestion control algorithms are a subset of congestion control algorithms designed to solve the scalability problem of the TCP Reno Friendly congestion control. Example of scalable congestion controls are DCTCP [4], TCP Prague [5], BBRv2 [6] and SCReAM [7]. These algorithms adapt the rate of the sender node proportionally to the number of congestion signals received. If no congestion signals are received by the sender node, the rate is increased. On the contrary, if any congestion happens, the rate is decreased, but not decreased as much as with classic congestion control. It is decreased proportionally to the fraction of the CE-marked packets received at the receiver node. Scalable congestion control introduces two control signals on average per RTT regardless of the flow rate [8]. Hence, it is possible to scale with high throughput while keeping accurate congestion control dynamic.

III. L4S

L4S [9] is a technology intended to ensure very low delay for the IP traffic. It exploits ECN and is based on the idea of marking packets as CE as soon as the queue delay in the network node starts to increase above a threshold. The threshold can change for different implementations. The sender reduces the rate due to the congestion signals, and thus it is possible to keep a low queue delay (and consequently lower end-to-end delay). At the same time, thanks to the use of scalable congestion control algorithms, it is possible to achieve a good throughput performance and high link utilization even if a high number of congestion events are signaled.

The steps of a communication session with the L4S support are given as follow:

- The sender indicates L4S support through the use of specific ECN codepoints (in this paper called L4S codepoints, which is explained in Table II).
- The network nodes recognize the packet as an L4S packet. When congestion occurs, it is signaled by changing the ECN bits to indicate CE.
- The packet reaches the receiver, and if the ECN bits show that congestion is experienced, the receiver notifies the sender about the congestion.
- The sender, once notified about the congestion, reduces the sending rate thanks to the scalable congestion control.

Certain conditions are required to achieve the L4S support. The first requirement concerns the endpoints (the sender and the receiver). The receiver must be able to notice the CE codepoint and notify the sender about the congestion.

Consequently, the sender modifies the transmission rate thanks to the scalable congestion control to prevent the network from congestion.

The second requirement concerns the network nodes. To effectively manage the congestion, each node in the network should be able to distinguish an L4S packet from a not-L4S packet (also called *classic* packet). The classic packets should be treated normally, while L4S packets should have a reserved queue where packets are CE marked as soon as the queue starts growing. For this reason, two different queues (one for the L4S capable traffic, and one for the not-L4S capable traffic) should be present in the network nodes. Hence, the suggested solution is the use of a Dual Queue Coupled AQM explained in [10].

As already mentioned, L4S exploits ECN, but uses the ECN bits in a different way with respect to the ECN standard [2]. The difference is found in the meaning of the ECN bits and codepoints, reported in Table II.

TABLE II: L4S codepoints and meaning.

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

In L4S, ECT(1) means that the traffic should be treated as L4S capable traffic, ECT(0) means that the traffic is ECN capable but not L4S capable (so it has to be sent to the *classic* queue), and the Not-ECT means that the traffic is not ECN capable (and thus not L4S capable). In a network node that has two different queues to serve classic and L4S traffic, packets with codepoints of Not-ECT and ECT(0) are sent to the classic queue, while packets ECT(1) and CE are sent to the L4S queue.

A. L4S Implementation Challenges in 5G New Radio

Although L4S is a technology proposed to fit in a general network context, there are challenges when implementing L4S in a specific network architecture. To implement L4S in a 5G network, two main challenges have to be addressed. These challenges are due to the presence of tunnels (GTP-U and IPSec) that can obstruct the propagation of the congestion signals, and the challenge related to the RLC layer that queued packets are already encrypted, and it is not possible to mark packets at the network nodes.

The challenge posed by the tunnels is solved by implementing network nodes and endpoints to be compliant with the IETF standard [11]. GTP-U and IPSec are IP-in-IP tunnels, a particular subset of tunnels where the tunnel's header is still an IP header. This means that the tunnel encapsulator has to copy the ECN bits of the incoming packet to the tunnel's header to propagate the congestion signal. Also, the tunnel decapsulator simply has to set the ECN bits of the inner packet accordingly to the tunnel's header. Thus, GTP-U and IPSec tunnels do not pose problems for the L4S signal propagation, and [11] explains how encapsulator and decapsulator should set the ECN bits, to correctly propagate the L4S signals. An example of how the decapsulator should behave is reported in Table III.

TABLE III: Tunnel decapsulator behavior for ECN propagation.

Arriving Inner Header	**Arriving Outer Header**			
	not-ECT	ECT(0)	ECT(1)	CE
not-ECT	not-ECT	not-ECT	not-ECT	drop
ECT(0)	ECT(0)	ECT(0)	ECT(1)	CE
ECT(1)	ECT(1)	ECT(1)	ECT(1)	CE
CE	CE	CE	CE	CE

To address the RLC layer challenge, two possible solutions can be applied. The first one is to move the queue from the RLC layer to Packet Data Convergence Protocol (PDCP) layer. However, this requires also the presence of an additional mechanism to send enough data to the lower layers to exploit all available physical resources. The second solution is to keep the queue at the RLC layer and mark packets at the PDCP layer. However, an additional mechanism is needed in this case to signal the level of the RLC queue to the PDCP layer. The second solution represents a sub-optimal case because tail marking is performed since the packets are marked at the PDCP layer, and thus the packets marked are the new incoming packets to the RLC queue. Tail marking means that the packets are marked from the tail of the queue. It is not an optimal solution because it involves a delayed signal of congestion. Packets marked have to wait for the prior packets already in the queue to be sent. The first solution has the problem that cannot be implemented in an already deployed 5G base station (gNB) since it involves a change in the gNB structure, while the second solution can be used in every gNB but with the drawback of not being an optimal solution. In this paper, we used the second approach, to assess the solution that can be used in every product with sub-optimal performance.

IV. L4S Implementation

A. Implemented L4S Marking Strategy

In this paper, the L4S marking strategy is implemented at the PDCP layer only in the gNB, which is regarded as the network node. A packet is marked as CE at the gNB according to a probability called marking probability $pMark$. The marking probability is updated periodically every dT ms and has three configuration parameters.

The first one is the Lower Threshold $l4sLowTh$ that represents the minimum threshold, below which a packet must never be marked. The Higher Threshold $l4sHighTh$ represents the maximum acceptable queue delay, above which a packet must always be marked. The range of $l4sLowTh$ - $l4sHighTh$ is called *threshold span*. There is also a parameter α that aims to give a memory to the process of updating the marking probability, thus avoiding peaks and obtaining a smoother behavior. In this paper, the second solution explained in Section III-A has been adopted, which is detecting the congestion level at the RLC layer, but marking packets at the PDCP layer.

The formula for calculating the marking probability at time t is composed of two steps, which are given as follows:

$$tmp(t) = \frac{qDelay(t) - l4sLowTh}{l4sHighTh - l4sLowTh}, \tag{1}$$

$$pMark(t) = \alpha \times tmp(t) + (1 - \alpha) \times pMark(t - dT). \tag{2}$$

The variable tmp represents the instantaneous marking probability at time t, computed through a linear probability function in (1). $tmp(t)$ is then used to calculate $pMark$ via the smoothing function in (2). $qDelay(t)$ represents the queue delay value at time t. $pMark(t - dT)$ is the last computed $pMark$, where dT is the time interval between two updates. The value for dT used in this work is 5 ms.

The parameter α in (2) weighs the contribution of the previous marking probabilities, and it belongs to the range $[0, 1]$. An α value equal to 1 implies that the marking probability updating process becomes memory-less since $pMark(t)$ will be equal to the instantaneous marking probability $tmp(t)$.

B. Simulation Scenario

The simulation scenario used to test L4S is created with the idea of getting as close as possible to the real case, and to have results with high external validity. Medium access control, RLC, and PDCP layers are modeled with high accuracy according to 3GPP specifications.

The simulation is performed with full NR and 5G Core Network (CN), however, the physical layer follows LTE specifications, with a carrier frequency of 600 MHz, a bandwidth of 10 MHz, and the frequency division duplex (FDD) transmission scheme. Furthermore, the CN is simplified since this study focuses on the bottleneck in the radio access network (RAN).

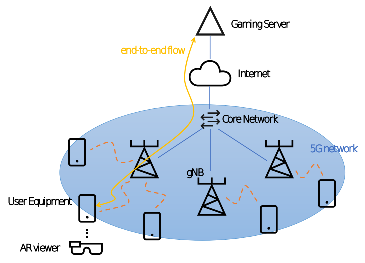


Fig. 1: Simplified Simulated Scenario.

Fig. 1 represents a simplified version of the simulated scenario. In particular, the simulation scenario consists of 7 gNBs with 3 sectors each, for a total of 21 cells. Each cell has 4 antennas transmitting on the base station side, while the user equipment has a single antenna. For each simulation, the simulation time is 100 s, and the first 20 s of simulation data is discarded to avoid initial instabilities affect the simulation results. The one-way latency between the sender, i.e., the server that provides the video content, and the base radio station is set to 25 ms. This value of latency is chosen to mimic the use-case where the video gaming server is relatively far from the end-user. It is an adverse scenario where the propagation of congestion signals and the reports of congestion is slower. This leads also to a slower response to the congestion at the endpoints.

The number of users running latency-critical interactive applications is fixed in each simulation. To further stress the network and get closer to simulating a real use case, web traffic users are also present in the system, in proportion to the number of video users. All the users are distributed randomly in the scenario, and they are served from the cell that has the best Reference Signal Received Power (RSRP). The users are not mobile and stay in their position during the whole simulation.

In Fig. 1, the Gaming Server represents the sender, while the receiver is the User Equipment, and they are responsible for the congestion control. The L4S marking is done at the gNB.

The number of video gaming users in the system is the parameter that is varied in simulations, and consequently, also the number of web users is variable, which is proportional to the number of video users. In particular, the number of video users can be 2, 5, 10, 15, 30 or 50. The proportionality factor chosen between video and web users is 10, which means that the number of web users in the system is 10 times the number of video users. For instance, when there are 2 video users in the system, there are also 20 web users. The inclusion of web traffic in the simulation setup provides a challenging environment that serves to test the performance of the gaming video under relatively harsh conditions. Besides, the relatively small carrier bandwidth (10 MHz) makes the impact on the individual even greater than for a larger carrier bandwidth.

In the created scenario, the only node in the network that has a queue, and thus where packets can be lost is the gNB. By default, the queues have unlimited memory and no AQM. This means that no packets will be marked or dropped from the network node, and consequently, the packet loss rate, by default, is zero. When the AQM is enabled, packets can be dropped, hence there is a possibility to have packet losses. The AQM policy consists in dropping only the oldest packet in the queue when the queue delay is above 100 ms. If the queue delay is above 500 ms, all the oldest packets are dropped until the queue delay return below 500 ms.

C. Key Performance Indicators (KPIs)

The **video frame delay** represents the delay experienced from the moment the video frame is generated by the video encoder at the gaming server to the moment it is received and recreated by the application at the receiver side. This delay can be decomposed into three smaller components:

$$VideoFrameD = RTPQD + ProcD + IPdelay, \tag{3}$$

where $RTPQD$ is the Real-time Transport Protocol (RTP) queue delay that is a component due to the SCReAM sender side architecture. A queue is present at the application layer of the sender side, where RTP packets carrying the multimedia traffic can wait to be sent. $ProcD$ is the delay due to regenerating the video frame from the application at the receiver side, and the $IPdelay$ is the layer 3 delay, i.e., delay of IP packets. It is the delay from the moment it is sent from the gaming server until the moment it reaches the final destination. Hence, IP delay can be decomposed in (4):

$$IPdelay = NetQD + PropD, \tag{4}$$

where $NetQD$ represents the network queue delay, which is the sum of the queue delay experienced in every node of the network, and $PropD$ is the propagation delay.

The **transmitted rate** is the application layer (or layer 5) throughput. It is computed as the amount of data transmitted in a specific time period from layer 5. Thus, it represents only the amount of data necessary for the application, excluding the overhead imposed by headers of lower layers. In this paper, *transmitted rate* and *throughput* are used interchangeably. The last KPI is the **packet loss rate** that represents the number of packets lost with respect to the number of packets sent.

D. SCReAM and Traffic Model

The scalable congestion control algorithm used in this work is SCReAM [7], a window-based and byte-oriented congestion control protocol intended for RTP traffic. It is a specific type of traffic that works on IP, designed to accommodate audio and video content. It is based on the self-clocking principle [12] that uses two separated mechanisms. The first mechanism works on a longer timescale and the second one works only in the last round trip time (RTT) that allows having a quick reaction to congestion and at the same time provides good throughput performances. Moreover, SCReAM implements a technique similar to the one proposed in [13] to estimate the network queue delay through the use of timestamps. SCReAM supports L4S, and it is designed to work with multimedia traffic especially in cellular networks.

Looking specifically at the traffic model, an *H.264* video encoder is used at the sender side. The encoder outputs video frames with a bitrate range set to $2-70$ Mbps. This range mimics the encoding at 4K resolution for the higher bitrates, with the ability to switch down to 1080p resolution for the lower bitrates. However, the video bitrate cannot go below 2 Mbps, as this would give a very poor video gaming experience. At the same time, this means that if the users have a throughput lower than 2 Mbps, the video frame will be queued in the RTP queue at the sender side, increasing the video frame delay.

As for the background traffic, web traffic is used to simulate real-case traffic, following 3GPP specifications [14]. The model first sends an index web page. The size of the page follows a normal distribution with a mean of 10710 bytes and a variance of 25032 bytes. After that, a number of web embedded objects are sent in parallel. The size of the embedded objects follows a normal distribution with a mean of 7758 bytes and a standard deviation of 126168 bytes. The number of embedded objects follows a Pareto distribution with a location of 15 and shape 3.1. The client reads the web page content for a random time that follows an exponential distribution with a mean of 5 seconds and a maximum reading time of 7 second. Afterward, the user requests a new web page, and this is done until the simulation stops.

E. Scheduling and Priority of Packets

In the gNB, we have one queue for every Protocol Data Unit (PDU) session (usually for every user). This means that if we have one cell with 5 users, in the gNB we have 5 different queues. Prioritizing one of these queues over another can have a major impact in terms of performance, especially with regards to latency. For this reason, two different schedulers are used to support the video gaming traffic. The two schedulers used are Round Robin (RR) and Delay Based Scheduler (DBS). RR is the simplest scheduler, that allocates resources once among the queues that have the same priority. DBS instead always schedules first the oldest packet among the queues, and it also assigns higher priority to the queues with packets older than a predefined threshold. In this paper, the DBS threshold is set to be equal to $l4sHighTh$. The reason for this choice is to keep the latency as low as possible: if the queue delay is above the $l4sHighTh$, then the packets are prioritized. For instance, assuming a DBS with a threshold of 10 ms, if among the video gaming users queues one packet is older than 10 ms, the whole queue is prioritized and the oldest packet is the first packet to be scheduled. When there are no more packets older than 10 ms in the queue, the queue came back to the default priority. RR instead, schedules packets of the first queue in slot 1, while slot 2 schedules packets from queue 2, and so on.

The real-time video is transmitted over RTP/User Datagram Protocol (UDP) with SCReAM as congestion control. Furthermore, by default, there is no special RLC buffer handling for L4S other than that ECN marking is added when queues build up. Quality of service (QoS) handling can be applied to give higher priority for the real-time media, e.g. in the case of the DBS scheduler.

V. Simulation Results

A. Preliminary Results

Firstly, we perform a simulation study to understand how α and the thresholds impact the KPIs. As can be seen from Table IV, increasing α means to consider more about the current queue delay rather than the history of it. It means that we do not consider the packets accumulated at the queues. Hence, as we decrease α, the throughput decreases while the network queue delay decreases. Table V shows the effect of the threshold span on throughput and IP delay. This shows that enlarging the threshold span means higher throughput but slightly increased latency. The results reported in Table IV and V refer to a simplified case with 3 cells only, using L4S with RR, but the same trend has been confirmed using the scenario described in Section IV-B.

B. Major Results

To assess the performance obtained with the use of L4S, simulations have been performed with and without L4S support. The results presented are obtained with $\alpha = 0.25$ and the threshold span $5-10$ ms.

TABLE IV: Impact of α.

α	**Throughput [Mbps]**	**Network queue delay [ms]**
0.5	17.0274	3.2016
0.25	16.9791	3.1712
0.125	15.9100	2.7426

TABLE V: Impact of the threshold span.

Threshold span	**Throughput [Mbps]**	**IP delay [ms]**
5-10 ms	31.31	26.9
7-15 ms	33.40	27

The first result shows the video frame delay with respect to the number of video users as in Fig. 2. Using L4S and a DBS involves the best results both on average and 99th percentile. When the number of video users increases in the system, L4S with RR instead shows a degradation in the performance, and results are comparable to the not-L4S case. The degradation in RR is not caused by an L4S malfunction, rather it is a behavior originated by the SCReAM sender structure. Indeed, at the sender-side application layer, SCReAM requires a queue for the RTP packet produced by the video encoder. In case the link throughput between the sender and the receiver is lower than the video bitrate produced by the video encoder, the RTP packets are queued up at the RTP queue, increasing also the video frame delay. The main role of L4S is to keep the network queue delay low. Hence, a metric that can show more clearly the L4S functioning is the IP delay. It considers only the delay experienced in the network, thus excluding the delay at the sender application layer.

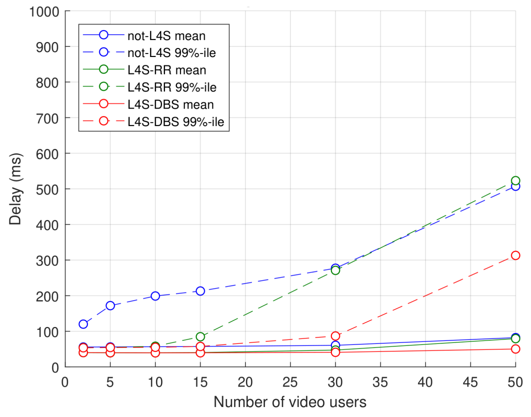


Fig. 2: Average and 99 percentile video frame delay vs. number of video users.

From Fig. 3, it is clear that L4S is improving the performance also in the RR case, both in average and also 99th percentile. If we look at the average behavior in the IP delay, the queue delay with RR and DBS is almost the same, and this is the confirmation that L4S is a technology that can work with every type of IP traffic regardless of the priority. In any case, as expected, the use of a scheduler that gives priority leads to better results (lower latency) if the 99th percentile is observed.

The other important aspect to consider is the transmitted rate (layer 5 throughput). Fig. 4 shows the transmitted rate of the AR video gaming users. As expected, using L4S involves a reduction in the transmitted rate performance. This is caused by the well-known throughput-delay trade-off, which makes it difficult to reach low latency and high throughput simultaneously for a flow. In any case, even at a high load in the system, the average transmitted rate performance is enough to provide a decent user experience since they are above 2 Mbps. Finally, with the use of DBS, it is possible to reach a higher average throughput performance.

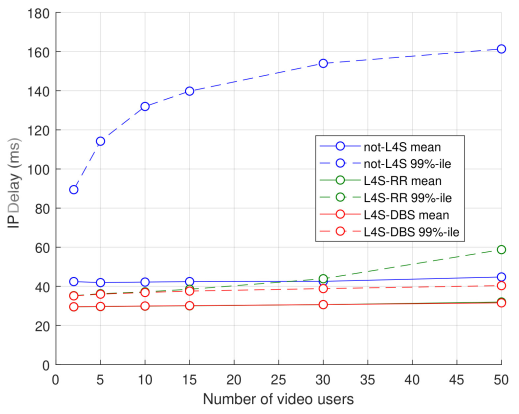


Fig. 3: IP delay vs. number of video users.

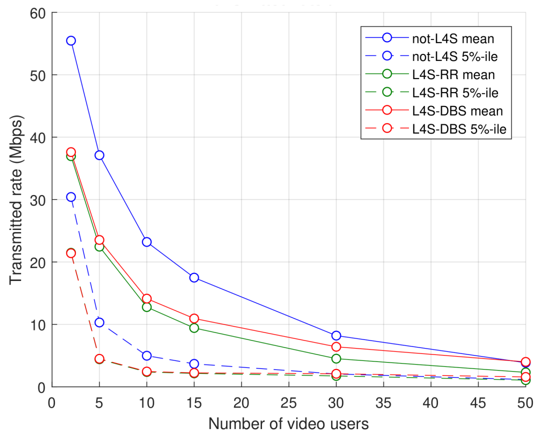


Fig. 4: Transmitted rate vs. number of video users.

The last parameter considered is the packet loss. In Fig. 5, the packet loss rate is reported, showing the percentage of packets transmitted that are lost in the network. The only point where the packets are lost is the queue at the PDCP layer in the gNB, where the AQM mechanism is implemented. As can be seen from Fig. 5, there is a clear improvement in the packet loss rate when L4S is enabled in the system.

When L4S is used in combination with a DBS, the average packet loss rate is zero, and also the 99th percentile is very close to zero. By using L4S with RR, there is an improvement of 8 times at high system load, both on average and 99th percentile.

The relation between throughput and latency is governed by an intrinsic trade-off. Higher throughput means higher

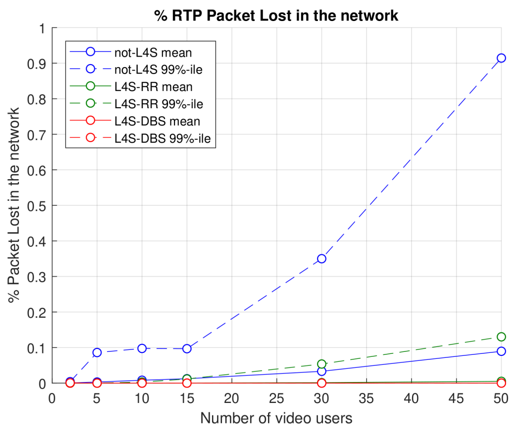
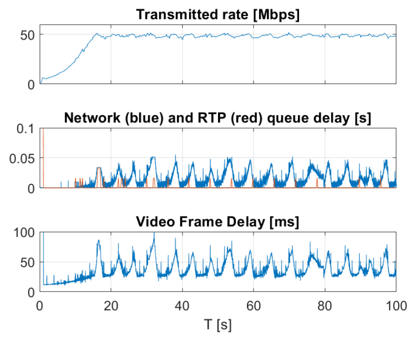
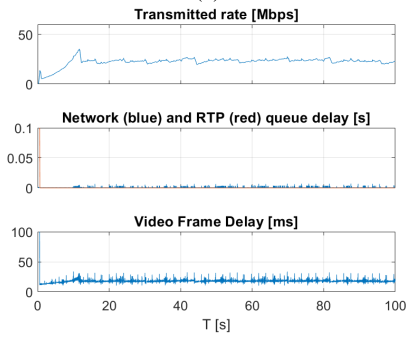


Fig. 5: Packet loss rate vs. number of video users.

latency, and vice versa, low latency implies a reduction in throughput. The throughput-latency trade-off is the explanation for throughput reduction when using L4S. It aims to achieve lower latency, but this entails a reduction in throughput. Fig. 6(a) shows the time course of the throughput, the network queue delay, the RTP queue delay at the sender side, and video frame delay values of a video gaming user. In this case, the scenario refers to a fixed communication network, where the capacity of the connection between the server and the client is 50 Mbps. For Fig. 6(b), $l4sHighTh$ is set to 10 ms and $l4sLowTh$ is set to 5 ms. Fig. 6(b) shows that L4S the throughput is reduced to about 25 Mbps. At the same time, there is a clear reduction in the RTP queue delay, network queue delay, and video frame delay with less variation. The



(a)



(b)

Fig. 6: Performance with (a) not-L4S vs (b) L4S.

sudden drop at $T = 10$ s in Fig. 6(b) is due to the fact that the SCReAM congestion control overshoots and adjusts down to the sustainable rate. One can also see that the video frame delay is slightly higher at $T = 10$ s.

VI. Conclusion

In this paper, Low Latency Low Loss Scalable Throughput (L4S) has been assessed as a technology to support a high-rate latency-critical application in a 5G network. The results show that using L4S, it is possible to achieve lower latency, lower packet loss and at the same time good throughput performance even at high load in the system. Moreover, if the L4S traffic is supported by a priority scheduler, all KPIs can be improved further. When L4S is used in combination with a Delay Based Scheduler (DBS), a packet loss rate very close to zero has been reached with a throughput of at least 2 Mbps, which provides decent performance for AR video game users.

References

[1] K. De Schepper, O. Bondarenko, I. Tsang, and B. Briscoe, "'Data Center to the Home': Ultra-Low Latency for All," RITE Project, Technical report, 2015. [Online]. Available: http://riteproject.eu/publications/
[2] K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," IETF, Tech. Rep. RFC 3168, 2001. [Online]. Available: https://tools.ietf.org/html/rfc3168
[3] F. Baker and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management," IETF, Tech. Rep. RFC 7567, 2015. [Online]. Available: https://tools.ietf.org/html/rfc7567
[4] S. Bensley, D. Thaler, P. Balasubramanian, L. Eggert, and G. Judd, "Data Center TCP (DCTCP): TCP Congestion Control for Data Centers," IETF, Tech. Rep. RFC 8257, 2017. [Online]. Available: https://tools.ietf.org/html/rfc8257
[5] B. Briscoe, K. De Schepper, O. Tilmans, M. Kuhlewind, J. Misund, O. Albisser, and S. A. A., "Implementing the 'Prague Requirements' for Low Latency Low Loss Scalable Throughput(L4S)," in *Proc. Netdev 0x13*, 2019. [Online]. Available: https://www.files.netdevconf.info/f/4d6939d5f1fb404fafd1/?dl=1
[6] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *ACM Queue*, vol. 14, September-October, pp. 20 – 53, 2016. [Online]. Available: http://queue.acm.org/detail.cfm?id=3022184
[7] I. Johansson and Z. Sarker, "Self-Clocked Rate Adaptation for Multimedia," IETF, Tech. Rep. RFC 8298, 2017. [Online]. Available: https://tools.ietf.org/html/rfc8298
[8] B. Briscoe and K. De Schepper, "Resolving Tensions between Congestion Control Scaling Requirements," Simula, Technical Report TR-CS-2016-001, 2017. [Online]. Available: https://riteproject.files.wordpress.com/2015/10/ccdi_tr.pdf
[9] B. Briscoe, K. De Schepper, M. Bagnulo Braun, and G. White, "Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service: Architecture," IETF, Tech. Rep. Draft ietf-tsvwg-l4s-arch-06, 2019. [Online]. Available: https://tools.ietf.org/html/draft-ietf-tsvwg-l4s-arch-06
[10] K. De Schepper, B. Briscoe, and G. White, "DualQ Coupled AQMs for Low Latency, Low Loss and Scalable Throughput (L4S)," IETF, Tech. Rep. Draft ietf-tsvwg-aqm-dualq-coupled-10, 2019. [Online]. Available: https://tools.ietf.org/html/draft-ietf-tsvwg-aqm-dualq-coupled-10
[11] B. Briscoe, "Tunnelling of Explicit Congestion Notification," IETF, Tech. Rep. RFC 6040, 2010. [Online]. Available: https://tools.ietf.org/html/rfc6040
[12] W. Hu and G. Xiao, "Self-clocking principle for congestion control in the internet," *Automatica*, vol. 48, pp. 425–429, Feb 2012.
[13] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind, "Low Extra Delay Background Transport," IETF, Tech. Rep. RFC 6817, 2012. [Online]. Available: https://tools.ietf.org/html/rfc6817
[14] "5G; Study on channel model for frequencies from 0.5 to 100 GHz," 3GPP, Tech. Rep. 3GPP TS 38.901, 2017.