IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 24, 2020, accepted February 23, 2021, date of publication March 4, 2021, date of current version March 16, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3063769

Preventing RLC Buffer Sojourn Delays in 5G

MIKEL IRAZABAL[1], ELENA LOPEZ-AGUILERA[1], ILKER DEMIRKOL[2], ROBERT SCHMIDT[3], (Graduate Student Member, IEEE), AND NAVID NIKAEIN[3]

[1]Department of Network Engineering, Universitat Politècnica de Catalunya, 08034 Barcelona, Spain
[2]Department of Mining, Industrial and ICT Engineering, Universitat Politècnica de Catalunya, 08034 Barcelona, Spain
[3]Department of Communication Systems, EURECOM, 06410 Sophia Antipolis, France

Corresponding author: Mikel Irazabal (mikel.irazabal@upc.edu)

This work was supported in part by the EU Horizon 2020 Research and Innovation Program under Grant 675806 (5GAuRA) and Grant 857201 (5G-Victori), and in part by the Secretaria d'Universitats i Recerca del Departament d'Empresa i Coneixement from the Generalitat de Catalunya under Grant 2017 SGR 376.

ABSTRACT The 3rd Generation Partnership Project (3GPP) is investing a notable effort to mitigate the endogenous stack and protocol delays (e.g., introducing new numerology, through preemptive scheduling or providing uplink granted free transmission) to attain to the heterogeneous Quality of Service (QoS) latency requirements for which the fifth generation technology standard for broadband cellular networks (5G) is envisioned. However, 3GPP's goals may become futile if exogenous delays generated by the transport layer (e.g., bufferbloat) and the Radio Link Control (RLC) sublayer segmentation/reassembly procedure are not targeted. On the one hand, the bufferbloat specifically occurs at the Radio Access Network (RAN) since the data path bottleneck is located at the radio link, and contemporary RANs are deployed with large buffers to avoid squandering scarce wireless resources. On the other hand, a Resource Block (RB) scheduling that dismisses 5G's packet-switched network nature, unnecessarily triggers the segmentation procedure at sender's RLC sublayer, which adds extra delay as receiver's RLC sublayer cannot forward the packets to higher sublayers until they are reassembled. Consequently, the exogenously generated queuing delays can surpass 5G's stack and protocol endogenous delays, neutralizing 3GPP's attempt to reduce the latency. We address RLC's related buffer delays and present two solutions: (i) we enhance the 3GPP standard and propose a bufferbloat avoidance algorithm, and (ii) we propose a RB scheduler for circumventing the added sojourn time caused by the packet segmentation/reassembly procedure. Both solutions are implemented and extensively evaluated along with other state-of-the-art proposals in a testbed to verify their suitability and effectiveness under realistic conditions of use (i.e., by considering Modulation and Coding Scheme (MCS) variations, slices, different traffic patterns and off-the-shelf equipment). The results reveal current 3GPP deficits in its QoS model to address the bufferbloat and the contribution of the segmentation/reassembly procedure to the total delay.

INDEX TERMS 5G, bufferbloat, low-latency, SDAP, RLC, OpenAirInterface.

I. INTRODUCTION

Ultra-reliable low-latency communications (URLLCs) are intended to address two orthogonal weaknesses faced in contemporary cellular networks: reliability and low-latency.

On the one hand, as Shannon proved in his information theory founding paper [1], given a noisy discrete channel and a transmission rate smaller than the channel capacity, there exists an encoding scheme capable of generating an equivocation rate (ϵ) arbitrarily small. This theoretical result shows how reliability depends on the coding scheme and confirms that new coding schemes (e.g., low-density parity-check (LDPC) codes) can achieve arbitrarily small equivocation rates at the expense of using big data blocks [2]. 3GPP has already defined the new channel codings [3], and the successful achievement of reliable communications is indispensable for the URLLC adoption. On the other hand, a large amount of efforts are being invested by 3GPP trying to mitigate or eliminate the latency introduced by endogenous cellular stack design (e.g., new numerology for mini-slots or pre-emptive scheduling [4]) and cellular protocol (e.g., uplink granted free transmission to avoid the Scheduling Request procedure

The associate editor coordinating the review of this manuscript and approving it for publication was Chung Shue Chen.

delay [5]). Moreover, a new sublayer for addressing different services has been introduced (i.e., Service Data Adaptation Protocol (SDAP) [6]), and a new QoS indicator (i.e., Quality of Service Flow Indicator (QFI)) will classify the different flows according to their different requirements [7]. However, exogenous 5G stack latency causes (i.e., latencies not directly induced by the 5G stack such as the bufferbloat or the RLC packet segmentation/reassembly procedure) can ultimately become the main contributors to the delay.

Bufferbloat at the RAN specifically occurs and plays a central role in the delay of low-latency traffic since (i) the data path bottleneck resides at the RAN as contemporary wireless channel capacity is inferior to wired channel capacity in contemporary networks; (ii) RANs are equipped with large buffers to absorb the unpredictable dynamic radio channel capacity and thus, avoid squandering wireless resources; and, (iii) flows with distinct characteristic share buffers on the data path in the 5G stack due to 5G's QoS funnel architecture. The first two premises combined with TCP's congestion control's greedy nature (e.g., TCP Cubic [8]) are necessary and sufficient to generate a plethora of packets at the bottleneck, which induces delays in the order of seconds [9]. In essence, packets from a greedy flow start accumulating at the bottleneck's link buffer, impeding a rapid packet delivery from other flows that share the same bottleneck queue. The ongoing bufferbloat research has primarily focused on the IEEE 802.3 and the IEEE 802.11 standards, achieving remarkable results [10] and proposing numerous new algorithms (such as Controlled Delay (CoDel) [11] and Fair Queuing CoDel (FQ-CoDel) [12]). The main challenge with regard to the bufferbloat is two-fold: (i) maintaining the buffer with enough data to fully use the available bandwidth and thus, avoid buffer starvation; and (ii) preventing excessive data in the buffer to minimize the packet's sojourn time.

3GPP has also introduced stack network improvements at the RLC sublayer in 5G [13] compared to 4G [14] aiming to decrease the latency. At 4G, the RLC Protocol Data Unit (PDU) header for data transmission consists of a fixed (i.e., one or two bytes depending on the configuration) and an extension part. The extension part is only present when more than one packet is assembled and is composed by an Extension bit (E) and a Length Indicator (LI). The E field is one bit long and indicates if another set of E and LI fields follows, or if the next bit is part of the data. The LI indicates the length of the packet and varies from 11 to 15 bits according to the configuration. In 5G the extension part has been renamed to Segment Offset (SO), and consists of 16 bits for all cases. It indicates the absolute position of the packet in bytes within the RLC PDU. This change of paradigm from a relative position to an absolute position (i.e., LI vs. SO) deteriorates the data compression ratio as the difference between packets starting positions is necessarily smaller than the absolute packets starting position. However, in modern processors the minimum amount of data that can be accessed is 1 byte. Therefore, if the information lies between 2 bytes, some bit manipulation assembly instructions need to be generated to access the value (e.g., with three E-LIs pairs of 12 bits, the information of the second packet starts at the bit $12 + 1 = 13$, which corresponds to the second byte, and ends at position $12 + 12 = 24$, which corresponds to the third byte). 5G simplifies this process as the packet starting position information is byte aligned and, thus, it can be directly accessed, sacrificing some throughput to reduce the latency. However, this latency reduction may not play an important role if another phenomenon that contributes to augment the delay in 5G is ignored: the segmentation/reassembly procedure. Every TTI, the MAC scheduler requests a PDU (i.e., an amount of bytes) from the RLC sublayer. This may involve segmenting an RLC Service Data Unit (SDU) packet to fit within the demanded total size of the PDU. If the packet is segmented, part of it is transmitted, while the rest waits at the sender's RLC sublayer. Once the rest of the packet is transmitted, a reassembly at the receiver's RLC sublayer occurs, and the packet is submitted to the Packet Data Convergence Protocol (PDCP) sublayer in the downlink procedure. Even though the segmentation/reassembly procedure depends on 5G stack exogenous causes (i.e., packet sizes, radio link conditions and MAC scheduler algorithm[1]), it has a non-negligible contribution in the delay that a packet suffers, as demonstrated in this paper. Moreover, slicing has emerged as a new 5G pillar feature [7], which fundamentally can be reduced to a resource allocation issue, and thus, exacerbates the problem. Unfortunately, recent research studies have mostly focused on resource distribution [15], [16], ignoring 5G's specificities and the packet-switched network nature of the Internet, and therefore, the delays associated with RLC's segmentation/reassembly procedure have been overlooked.

This paper addresses the bufferbloat and the segmentation/reassembly procedure in 5G, and in summary makes the following contributions:

- We analyze 5G's exogenous delays that arise at the RLC sublayer's buffers (i.e., bufferbloat and segmentation/reassembly procedure).
- We propose an enhanced 3GPP QoS scenario for improving the latency in 5G.
- We introduce a novel bufferbloat avoidance algorithm (i.e., e5G-BDP) based on the bandwidth delay product, which is the optimal theoretical pacing rate for avoiding the bufferbloat, while fully utilizing the link [17].
- We present a resource scheduling algorithm (i.e., Enhanced Quantum Partition (EQP)) considering 5G specificities that minimizes RLC's segmentation/reassembly procedure.
- We implement and evaluate the performance of our proposed algorithms (i.e., e5G-BDP and EQP) against current state-of-the-art solutions, emulating real network conditions and using off-the-shelf equipment, ultimately

[1]3GPP does not define the MAC scheduler algorithm, and therefore, it cannot be considered an endogenous stack delay cause.

validating our solutions as they significantly reduce the RLC buffer sojourn delay.

The rest of this paper is structured as follows. In Section II, 5G's contemporary QoS scenario is presented along with our enhanced 3GPP QoS architecture proposal. Section III thoroughly discusses the bufferbloat and the segmentation/reassembly procedure in 5G and describes related works in the field. Our proposed solutions appear in Section IV, while the utilized evaluation framework is described in Section V. In Section VI the results from our solutions and the state-of-the-art solutions are evaluated, and lastly, in Section VII we expose the conclusions of this paper.

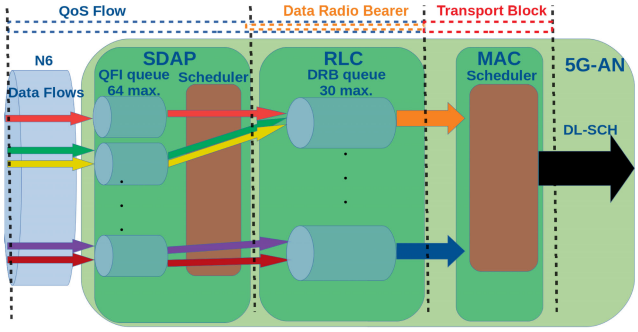

FIGURE 1. Proposed enhanced QoS model per UE.

II. BACKGROUND IN 5G's QoS MODEL

5G use case heterogeneity inherently creates a non-trivial QoS scenario that is described in [7]. In the following, the most important aspects of the QoS scheme for the data path in the 5G Access Network (5G-AN) are presented, assuming the downlink procedure unless otherwise mentioned.

Packets from the data network will flow through the N3 interface to the 5G-AN already marked with a QoS Flow Identifier (QFI) [18]. The QFI is responsible to denote among other things: the maximum data burst volume, the resource type, the packet priority level for scheduling purposes, the tolerated delay referred to as the Packet Delay Budget (PDB) or the tolerable error rate through the Packet Error Rate (PER). PDB indicates the upper bound for the permissible delay, measured from the N6 interface (i.e., from the moment that the packet arrives to the User Plane Function (UPF)) until the packet is received by the UE, while PER is defined as the amount of packets received in the UE's PDCP sublayer divided by the amount of packets forwarded by the RLC sublayer of the 5G-AN. Three different resource types are described by 3GPP: Delay-Critical Guaranteed Bit Rate (Delay-Critical GBR), Guaranteed Bit Rate (GBR) and Non-Guaranteed Bit Rate (Non-GBR) [7]. The first entity to apply QoS traffic engineering techniques in 5G-AN, is the newly defined SDAP sublayer [6]. An SDAP entity per PDU session is foreseen in [6], although it is mentioned that other implementations are valid. SDAP's main function or *raison d'être* is mapping the QFI flows into Data Radio Bearers (DRBs), according to the configuration provided by the Radio Resource Control (RRC). Therefore, it lacks any queue or scheduling capability, contrary to what is depicted in Fig. 1. However, the QFI is a 6 bit field (i.e., $2^6 = 64$ [19]), while the maximum number of DRBs is 30 [19]. For every DRB a new RLC entity is instantiated and therefore, an RLC buffer per DRB exists. This inevitably generates a funnel, as shown in Fig. 1, where a many-to-one relation will occur following the pigeonhole principle. 3GPP has not explicitly defined any scheduling capabilities in the SDAP. However, if stringent and diverse QoS requirements must be met, mobile network operators will need to provide a packet based scheduler, since finding and arbitrarily dequeuing packets with different QFIs that are already queued is costly and complex. Furthermore, if no scheduling capabilities are added to the SDAP sublayer, the only sublayer at the 5G-AN where QoS traffic engineering techniques can be applied is the MAC, which will pull data from bloated RLC buffers if the packets are forwarded as they arrive. Until the RLC sublayer, data is transmitted in packets, where a header is added to the original packets that arrived to the SDAP sublayer. However, once the MAC sublayer notifies the RLC sublayer about the amount of bytes that need to be forwarded, packets are joined to form the requested Transport Block (TB). If the requested TB size (TBS) cannot be filled with the current packets (e.g., two 1000 bytes packets in the RLC buffer, yet the MAC requests 1500 bytes) a packet segmentation occurs. Consequently, part of the original packet is transmitted to the UE, while the rest waits at the 5G-AN. This phenomenon delays the information delivery, as packets cannot be submitted to the UE's PDCP until reassembled. Lastly, the MAC scheduler pulls the required data from the RLC queues and forwards it through the Downlink Shared Channel (DL-SCH), as observed in Fig. 1. To gain finer control over the packets' QoS, we enhance the current SDAP standard with two new capabilities: (i) we add a queue per QFI (i.e., 64 queues per SDAP entity) with the purpose of segregating different traffic flows and retain the packets at the SDAP sublayer, and (ii) we provide scheduling capabilities to the SDAP sublayer.

III. PROBLEM DESCRIPTION AND RELATED WORK

This section describes the two phenomena analyzed in this paper along with the existing related work. In the first subsection, the bufferbloat phenomenon at contemporary cellular networks along with the state-of-the-art solutions for addressing it are presented. In the second subsection, RLC's segmentation/reassembly procedure is thoroughly studied and its effect on augmenting the delay is exposed.

A. BUFFERBLOAT AT THE RLC SUBLAYER

1) PROBLEM DESCRIPTION

Bufferbloat is the name by which the effect of excessive buffering of packets in the bottleneck's data link buffer is known. Such effect in the 5G stack is caused by (i) sender's congestion control algorithm (e.g., TCP Cubic [8]), and

(ii) large buffers at the bottleneck link (i.e., RLC sublayer buffers). The default Linux kernel congestion control algorithm (i.e., TCP Cubic) is loss-based. Therefore, the congestion is detected through a packet lost, which ideally coincides with the available bandwidth.

However, if large buffers are deployed at the bottleneck's link, a queue is formed once the available bandwidth is reached, misinforming TCP's congestion control algorithm, as the packets are not lost, but rather experience a larger delay than expected due to the sojourn time at bottleneck's link. In essence, TCP's congestion control algorithm cannot differentiate between the delay generated by congested buffers, and the delay produced by the packet propagation. Contemporary wireless links are considerably slower than wired links (e.g., a 20 MHz bandwidth LTE base station can forward approximately 70 Mbit/s with a 28 MCS [20], in contrast with 400 Gbit/s of an IEEE 802.3db fiber-optic physical media interface), forming the bottleneck at the RAN, and specifically at the RLC sublayer, where the last buffer before the wireless transmission is located. Due to the dynamic nature of the radio channel capacity, service providers deploy large RLC buffers aiming to avoid squandering the scarce wireless resources, and thus, unintentionally generating the necessary conditions for the bufferbloat to appear. The challenge to avoid the bufferbloat in 5G is explained in the following dichotomy. On the one hand, the RLC buffer must contain enough bytes to feed the MAC sublayer. A failure to this requirement results in wireless resource under-utilization. On the other hand, no more bytes than the requested from the MAC sublayer should be waiting at the RLC queue, so that a low-latency flow can avoid unnecessary queuing sojourn time.

2) STATE-OF-THE-ART SOLUTIONS

Small queue sizes result in lower latencies as demonstrated in [21]. The basic idea is to reduce the amount of packets in the queues caused by overdimensioned buffers, while not starving the transmission channel. Due to the queue size limits, packets start accumulating at higher sublayers following a phenomenon known as back-pressure. If the packets reside at different queues, a scheduler can easily pull the most stringent demanding packet from the higher layer queues, reducing the sojourn delay. Such a scheme is used by Dynamic RLC (DynRLC) [22] and Enhanced Bearer Buffer (EBB) [23], presented by the same authors. These methods estimate the available bandwidth measuring the packet's sojourn time. If the sojourn time increases, the allowed size of the buffer, which is defined as the maximum number of SDUs, decreases and no more packets from higher sublayers (i.e., PDCP) are delivered. If on the contrary, the sojourn time decreases, the buffer capacity limit of the RLC is augmented. The Dynamic RLC Queue Limit (DRQL) [24] is a similar solution based on limiting the queue size, considering the amount of bytes instead of the number of SDUs remaining on the queue.

Another well studied policy for improving the buffer sojourn time is Active Queue Management (AQM) [25], being CoDel [11] the most widely applied AQM policy. CoDel is governed by two parameters: the desired delay (5 *ms* by default) and the interval time value (100 *ms* by default). A timestamp is added to every newly ingressed packet and the sojourn time is measured when packets egress. If the sojourn time of all the packets egressed during the interval time have been above the desired delay value, the next packet is dropped, informing the sender that congestion is happening, and the control law that determines the next drop time is updated. The next drop time is reduced in inverse proportion to the square root of the number of packets dropped since the dropping state was entered. Such an approach permits the existence of bursty traffic during periods shorter than the interval value. Recent results on CoDel in cellular networks [21], [24] [26] show a latency reduction when adopted.

Other state-of-the-art solutions try to avoid the bufferbloat through the *keep the pipe justfull*, *but not fuller* principle described by Kleinrock [17]. The TCP BBR [27] algorithm fulfills such principle from the OSI Layer 4 perspective. BBR estimates the actual bandwidth observing the Round Trip Time (RTT) of the packets, and it interprets an increase in the RTT as a sign of bottleneck forming, thus reducing its sending rate. However, as demonstrated in [24], such an approach may not be optimum in a mobile network. A similar approach is considered in [28], where the congestion control algorithm is manipulated through the ECN bits to accelerate or slow down the packet delivery rate. Other advanced algorithms that are based on the same principle within the cellular network are 5G-BDP and USP [24], which reported promising results.

Segregating the flows into different buffers is one of the solutions that can partly eliminate the sojourn time induced by other flows. It has been successfully implemented in the Fair Queuing scheduler [29], and presents many variants (e.g., FQ-CoDel [12]). A greedy flow will not monopolize a queue's assigned throughput as it rests in a separate queue, and a scheduler pulls data according to different policies (e.g., round-robin or earliest deadline first). Recently, a new research paradigm (i.e., Low Latency, Low Loss, Scalable Throughput (L4S) [30]) proposes to segregate the packets into two queues: one for the traffic prone to generate queuing delay, and the other one for traffic that inherently avoids the bufferbloat (e.g., traffic generated through TCP's BBR congestion control algorithm). However, there exist many services that demand stringent low-latency delays to function correctly (e.g., VoIP [31] or mobile gaming [32]), where identifying and tagging them for segregation purposes represents a challenge, due to their origin's dynamic nature (i.e., servers can be relocated, altering their 5-tuple identifier) and amount. Furthermore, the rising digital privacy concerns in contemporary societies will increase the encrypted, as well as, relayed traffic, disabling deep packet inspectors and origin/destination identifiers' tagging capability. Lastly, as explained in Section II, the 5G QoS architecture forms a funnel with limited QFIs (i.e., 64) and DRBs (i.e., 30) per UE, and therefore, the solution of segregating the packets lacks scalability as the number of flows grows.

B. SEGMENTED PACKETS AT THE RLC SUBLAYER

1) PROBLEM DESCRIPTION

Packets shall flow through the 5G stack until the RLC buffer, as shown in Fig. 1 and may start accumulating there, as the wireless link is the slowest link in the data path. Packets wait at the RLC sublayer until the MAC scheduler pulls a specific number of bytes. While every UE will be provided with at least one DRB, up to 30 DRBs per UE can coexist, and therefore, 30 RLC buffers. Parallel queues will be formed, and thus, the MAC scheduler has to map the available resources to the RLC buffers according to the scheduler policy (e.g., round-robin). The RLC buffer is a FIFO queue [33], [34] since packets within a DRB should be treated equally [7], and, therefore the MAC scheduler can only access the most recently inserted packets after it has pulled the older ones. This introduces a precedence relation for the packets that belong to the same queue (i.e., packets from the same queue can only be egressed following the arrival order). Furthermore, the radio resource allocation performed by the MAC scheduler does not map directly bytes to the RLC buffers, but rather it assigns Resource Blocks (RBs). RBs are grouped into Resource Block Groups (RBG) according to the cell configuration (e.g., in a 5 MHz bandwidth cell with Type 0 and Configuration 1, the minimum number of RBs to distribute is 2, except for the last RB [35]), which forms the smallest unit that can be assigned to a UE. Moreover, to assure an error rate below 10% [35], the UE delivers a channel quality estimation through the Channel Quality Indicator (CQI) in uplink. The CQI is a scalar and its value is translated into a Modulation and Coding Scheme (MCS) [35]. The MCS defines the modulation to use (i.e., BPSK, QPSK, 16 QAM, 64 QAM or 256 QAM that transmit 1, 2, 4, 6 or 8 bits per symbol, respectively) and coding rate, and thus, the channel capacity is determined by the radio conditions (i.e, under good radio conditions, larger amount of information can be transmitted).

Furthermore, 3GPP defines three different modes in which a RLC entity can be instantiated [13]: Transmission Mode (TM), Unacknowledged Mode (UM) and Acknowledged Mode (AM). Through a TM entity, only control information can be forwarded, while data information can flow by either a UM or AM entity. Both UM and AM share the ability to segment a packet if the TBS notified by the MAC does not fit within the size of the packets waiting, as seen in Fig. 2. According to 3GPP [13], packets at the RLC sublayer will be segmented if the RLC SDU size is larger than the bytes requested by the MAC sublayer. As seen in Fig. 2, once packets are segmented and a RLC header is added, they are transmitted to the receiver's RLC, where after removing the RLC header, they wait for a SDU reassembly before submitting them to the next sublayer (i.e., UE's PDCP in the downlink procedure). Therefore, information will not be forwarded until a complete reassembly occurs, which in the best case will occur in the next TTI. Segmentation/reassembly procedure guarantees a full frequency spectrum utilization in the cases where the next packet size exceeds the TBS.

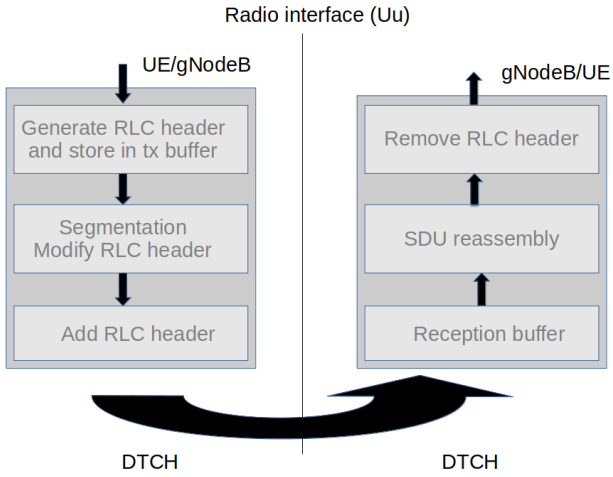

FIGURE 2. RLC UM functions as described by 3GPP [13].

For example, in a static scenario with a LTE base station with a 5 MHz bandwidth, under the best channel conditions (i.e., 28 MCS), approximately 2289 bytes can be transmitted every TTI [20]. However, bulky flows in an IP network will use the maximum allowable packets size (i.e., 1500 bytes in Ethernet) to minimize the protocol's overhead and maximize the transmitted information ratio. This example shows that even ignoring the dynamic radio link channel's capacity (i.e., assuming a static TBS of 2289 bytes), a myriad of fragmented packets at the RLC sublayer are generated as the TBS notified by the MAC would rarely coincide with the packets' size, and consequently, the delay is increased.

The principal constraints to consider in RLC's segmentation/reassembly procedure can be summarized as:

- The RLC buffers are FIFO queues, and thus the packets cannot be pulled arbitrarily.
- The resource allocation is performed through RBG, rather than bytes.
- The MCS determines the channel capacity, which dynamically changes according to the radio link conditions.

2) STATE-OF-THE-ART SOLUTIONS

Resource allocation has recently received significant attention as it is one of the pivot ideas around the slicing concept [36], [37] [38]. Unfortunately, most literature about slicing discusses the resource allocation problem without considering 5G's packet-switched network nature (e.g., segmentation problem where the information is not forwarded unless all the fragments are reassembled), or 5G's RB distribution peculiarities (e.g., RBG).

In contrast, in other protocols such as Ethernet (i.e., IEEE 802.3) or Wi-Fi (i.e., IEEE 802.11), the data is transmitted asynchronously. In Ethernet, the MTU size is

1500 bytes[2] [39], while in Wi-Fi it reaches 2304 bytes [40]. Even though modern versions of Ethernet do not use any collision detection mechanism, Wi-Fi employs a collision avoidance mechanism to orchestrate the access of multiple stations to the wireless channel. An acknowledgment frame from the receiver is used to confirm that the data arrived correctly. IEEE 802.11 increases its probability of successfully transmitting a packet in a noisy channel through packet fragmentation. However, the asynchronous nature of accessing the channel, compels to aggregate frames for achieving full bandwidth [41], and therefore, the fragmentation procedure is rare in comparison with the RLC segmentation/reassembly mechanism that arises in cellular networks due to their synchronous nature. Therefore, the segmentation/reassembly procedure can be mostly considered as a cellular network specificity, and consequently, it has not been thoroughly researched in IEEE 802.3 and IEEE 802.11 standards.

IV. PROPOSED SOLUTIONS TO ADDRESS 5G's EXOGENOUS DELAYS

In this section, the proposed solutions to mitigate the exogenous delays suffered by the 5G network stack are presented. Specifically, a bufferbloat avoidance algorithm is presented along with a novel algorithm that reduces the RLC SDU segmentation/reassembly procedure delay.

A. ENHANCED 5G-BDP (e5G-BDP)

In [24], we presented different solutions that rely on the communication between the SDAP and the MAC sublayers. Among them, 5G-BDP was presented and shown as the most successful algorithm to address the bufferbloat. 5G-BDP calculates the Bandwidth Delay Product (BDP), which is the optimal pacing value to forward the packets from the SDAP to the RLC to work on the optimal rate [17]: not starving the MAC scheduler, but also avoiding to bloat the RLC buffer.

However, 5G-BDP expects a uniform bandwidth scheduling between TTIs (i.e., it expects the MAC scheduler to distribute the RBs uniformly, such as {6,6,6,6,6} instead of {0,12,0,0,18} during 5 TTIs). The resource distribution relies on the MAC scheduler policy, from which 5G-BDP is decoupled, as it only acquires RLC buffer occupancy information. For example, the MAC scheduler algorithm may assign RBs according to the current buffer occupancy. In such scenarios, where the RBs are not uniformly distributed, a non-virtuous cycle may occur in 5G-BDP. 5G-BDP will not forward packets to the RLC sublayer, since in the last TTI no RBs were assigned to the RLC buffer, while the MAC scheduler might not assign more RBs to the RLC buffer due to its low occupancy. Additionally, 5G-BDP does not consider the size of the current packet to submit. Once a large packet is queued in the RLC buffer, completely submitting it can last several milliseconds, especially in scenarios with low throughput, and therefore, to avoid forwarding them unless necessary also improves the sojourn time suffered by low-latency packets.

To avoid the situations described, we present the *enhanced 5G-BDP (e5G-BDP)* solution which is threefold. In the first place, the BDP is calculated through an Exponentially Weighted Moving Average (EWMA). An EWMA smooths out short term fluctuations while exposes longer term trends. This provides a more accurate bandwidth computation and absorbs outlier bandwidth oscillations (e.g., bandwidth variations due to HARQ/NACK or non-uniform scheduling). In the second place, packets are forwarded more actively in comparison with 5G-BDP, avoiding the non-virtuous cycle previously explained. In e5G-BDP packets are also forwarded according to the BDP, yet ignoring the amount of accumulated bytes at the RLC buffer. However, if packets are not forwarded in the following TTI, the BDP will be reduced, and thus this second measurement can be thought as a smooth packet forwarding reduction mechanism. In the third place, the size of the packet to forward to the RLC buffer is considered (i.e., small size packets are more proactively submitted to the RLC buffer while large packets tend to stay longer at the SDAP sublayer).

e5G-BDP works within a 1 *ms* TTI, as it is the lowest common denominator of 5G's possible TTIs, since the slot duration in 5G fluctuates between 1 *ms* and 62.5 μs at the expense of using more frequency spectrum [42].

The pseudo-code of e5G-BDP is presented through the Algorithms 1, 2 and 3. e5G-BDP algorithms are executed by different sublayers (i.e., SDAP and MAC). Algorithm 1 represents the main e5G-BDP function, which is executed by the SDAP scheduler per active RLC buffer periodically within a TTI, to determine whether a packet can be sent to the subsequent lower sublayer. e5G-BDP calculates the BDP per RLC buffer. It first checks whether the *update_flag* variable was set (line 1) by the MAC scheduler, and if so, it recalculates the bandwidth calling the function *update_last_bandwidth* given in Algorithm 3. The *update_flag* is set every 1 *ms* by the MAC sublayer, after the packets from the RLC sublayer are received. At line 5, it is checked if the sum of the bytes submitted from the SDAP to the RLC (i.e., SDAP-RLC submitted bytes *srs_bytes*) and the last accumulated bytes in the RLC buffer surpasses the maximum capacity under the current MCS. If the sum surpasses the capacity, a true value is returned, thus, informing the SDAP scheduler that the limit is reached, and therefore, no more packets are forwarded to the RLC buffer. If the sum is smaller than the capacity, the *paced_bytes* (line 8) are calculated. This feature, enables the pacing capability, as the *paced_bytes* depend on the elapsed time since the last 1 *ms* TTI (e.g., if the actual bandwidth is 2200 bytes/TTI and the elapsed time since the last 1 *ms* TTI is 0.5 *ms*, 1100 bytes are the theoretical paced bytes, without any sum or multiplicative factor). In this manner, large packets (e.g., 1500 bytes packets) are more likely submitted during the last moments of the TTI (e.g., (750, 1000) μs interval), and therefore, if a low-latency packet with higher priority arrives at the beginning

[2]The standard explicitly talks about octets rather than bytes. Through this paper, no differentiation is made and the bytes are considered as 8 bits objects or octets.

of a TTI (e.g., (0, 750) μs interval) it can avoid a bloated buffer. It can be argued that the best results could be achieved if the packets are kept at the SDAP sublayer and forwarded to the RLC sublayer just a moment before the TTI (i.e., TTI^{-}). However, the cellular network is a real time system, where the lower layers/sublayers (i.e., PHY layer or MAC sublayer) have higher priority than the higher sublayers (i.e., RLC, PDCP or SDAP sublayers). Therefore, it may occur than the TTI^{-} forwarding opportunity at the SDAP sublayer is missed, which would lead to starve the MAC sublayer (i.e., not having enough bytes to forward from the RLC to the MAC), and thus, squander throughput. At line 9, the *paced_bytes* is reduced by a constant in the range of (0.0, 1.0], and if it exceeds the sum of the next packet data size (i.e., size of the packet candidate to forward to the RLC from the SDAP) and the already submitted bytes, a false value is returned, informing the SDAP scheduler to submit the packet. This feature enables the forwarding of the packets independently of the last measured buffer occupancy, contrarily to what happens at line 16, and mostly in the latter moments of the TTI (i.e., in the (500, 1000) μs range rather than the (0, 500) μs range since last transmission, assuming a 1 *ms* TTI), as the *paced_bytes* value increases with the elapsed time and, therefore, dispatches packets more actively as compared with 5G-BDP. It also favors forwarding small packets. At line 12, the *extra_bytes* variable is initialized to *data_size*/5, and if a packet has already been transmitted and some bytes were accumulated during the last TTI, it is set to *data_size*/3 at line 14. It incentivizes forwarding the next packet if during the current TTI no packet has already been forwarded and the RLC buffer was emptied in the last TTI. This variable also discourages submitting large packets that are one of the causes of the bufferbloat. Lastly, at line 16, the already submitted bytes (i.e., *srs_bytes*) plus the last accumulated bytes on the buffer, and the *extra_bytes* variable, are compared against the *paced_bytes* to decide to forward or keep the packet at the SDAP sublayer. Contrarily to line 9, in this last condition (i.e., line 16) the occupancy of the RLC buffer after submitting the RLC PDU is considered before forwarding the packet.

Algorithm 2 and Algorithm 3 are helper functions, where the last bandwidth according to the RLC buffer occupancy after the 1 *ms* TTI is estimated, and the *paced_bytes* are calculated. In Algorithm 2, the bandwidth is estimated according to the number of bytes that were pulled by the MAC from the RLC buffer through an EWMA calculation. The bytes that remained in the RLC buffer are gathered (i.e., line 1), the bytes that were transmitted to the MAC sublayer are calculated (i.e., line 2), and the new bandwidth is estimated (i.e., line 3). Lastly, the submitted bytes from the SDAP sublayer to the RLC are reset (i.e., line 4), and the *last_acc_bytes* and *last_tti* updated. As previously mentioned, e5G-BDP is intended to be executed per active DRB, and thus, Algorithm 2 is called for each active RLC buffer. In Algorithm 3 the *paced_bytes* are calculated, based on the elapsed time, the bandwidth and the last accumulated bytes.

Algorithm 1: e5G-BDP *limit_reached*. It Answers the SDAP Sublayer Whether to Keep a Packet or Forward It

Input: Size of the next packet (*data_size*)
Output: Bool value whether to forward or keep the packet

1: **if** *update_flag* == *True* **then**
2: *update_bw_est*(); // Alg. 2
3: *update_flag* = *False*
4: **end if**
5: **if** *srs_bytes* + *last_acc_bytes* > *max_bytes* **then**
6: **return** *True*;
7: **end if**
8: *paced_bytes* = *calculate_paced_bytes*(); // Alg. 3
9: **if** *paced_bytes* × *reduce_const* > *data_size* + *srs_bytes* **then**
10: **return** *False*;
11: **end if**
12: *extra_bytes* = *data_size*/5;
13: **if** *srs_bytes* ≠ 0 ∧ *last_acc_bytes* ≠ 0 **then**
14: *extra_bytes* = *data_size*/3;
15: **end if**
16: **if** *srs_bytes*+*last_acc_bytes*+*extra_bytes* > *paced_bytes* **then**
17: **return** *True*
18: **end if**
19: **return** *False*

Algorithm 2: e5G-BDP *update_bw_est*. It Estimates the Actual Bandwidth per RLC Queue

Input: Current time (*now*)
Output: Updated bandwidth and RLC buffer occupancy information (*rms_bytes*,*bandwidth*, *srs_bytes*, *last_acc_bytes* and *last_tti*)

1: *acc_bytes* = *get_bytes_rlc_queue*();
2: *rms_bytes* = *srs_bytes* − (*acc_bytes* − *last_acc_bytes*);
3: *bandwidth* = *EMWA*(*rms_bytes*/*TTI*);
4: *srs_bytes* = 0;
5: *last_acc_bytes* = *acc_bytes*;
6: *last_tti* = *now*;

A multiplicative factor variable at line 2 (i.e., *incr*) that depends on the elapsed time helps forwarding packets more actively during the latter moments of the TTI, since the value increases during the second half of the TTI. Therefore, the packets are forwarded more actively during the last moments of the TTI, avoiding a possible starvation that would have an impact in the throughput. If the bandwidth is not equal to zero, an additive factor of *MTU*/7 is used, again, to avoid a possible starvation in real time systems at line 5. The value of *MTU*/7 is chosen as a compromise in a packet based network as 5G, where most of the packets will be based on TCP and transported through Ethernet

Algorithm 3: e5G-BDP *calculate_paced_bytes*. It Calculates How Many Bytes Could Had Been Forwarded Theoretically, According to the Current Bandwidth and the Elapsed Time Since Last TTI

Input: Current time (*now*)
Output: Paced bytes (*paced_bytes*)

1: $elapsed = (now - last\_tti)/TTI$;
2: $incr = elapsed < 0.5?1.2 : 1.33$;
3: $paced\_bytes = 0$;
4: **if** $bandwidth \neq 0$ **then**
5: $paced\_bytes = incr \times elapsed \times bandwidth + MTU/7$;
6: **else if** $last\_acc\_bytes == 0 \wedge elapsed > 0.5$ **then**
7: $paced\_bytes = MTU/4$;
8: **end if**
9: **return** *paced_bytes*

(i.e., MTU of 1500 bytes), and UDP packets, that are normally scarce, small in size and do not contribute as severely to the bufferbloat as the TCP ones. The MTU value is divided by 4 (i.e., line 7) with the intention of forwarding the packet to the RLC buffer in the cases where the bandwidth was 0, the RLC buffer is empty and the elapsed time is beyond 50% of the TTI, as seen in line 6. This assures that a packet will be forwarded from the SDAP as the theoretical calculated value (i.e., MTU/4) will be bigger than the *data_size*/5 for all the packet sizes, as seen in Algorithm 1 at line 16.

e5G-BDP pacing mechanism is illustrated in Fig. 3. Let us assume that 600 bytes remained in the RLC buffer from the previous TTI (i.e., $last\_acc\_bytes = 600$ bytes), the calculated bandwidth is 1000 bytes/TTI, the MTU is 1500 bytes and that packets from two different flows share the RLC buffer, one bulky (i.e., 1500 bytes packets) and one with low-latency requirements (i.e., 200 bytes packets) [31] for a quantitative discussion. Let us assume that at $t = 1/4$ of the TTI since the last RLC to MAC forwarding event, e5G-BDP obtains an opportunity to forward packets from the SDAP sublayer to the RLC sublayer. These opportunities cannot be precisely predicted, as the upper stack sublayers (i.e., RLC, PDCP and SDAP) may miss some events, in contrast with the lower layers/sublayers (i.e., PHY and MAC), where synchronization is mandatory, and thus, real time predictability. As observed in Fig. 3, the packet to forward is large in comparison with the current computed bandwidth, which is calculated through an EWMA filter. The *paced_bytes* value is 1.2×0.25 *TTI* $\times$ 1000 *bytes*/*TTI* + 1500/7 *bytes* = 514 bytes, while the sum of the *srs_bytes* (i.e., bytes forwarded from the SDAP to the RLC during this TTI, 0 bytes), the *last_acc_bytes* (i.e., 600 bytes) and the *extra_bytes* (i.e., 1500/5 = 300 bytes) reaches 900 bytes. Therefore, e5G-BDP refuses to submit the packet, and the same reasoning applies when e5G-BDP obtains a forwarding opportunity at $t = 1/2$ (i.e., $paced\_bytes = 1.2 \times 0.5$ *TTI* $\times 1000$ *bytes*/*TTI* $+ 1500/7$ *bytes* = 814 bytes), since forwarding a large packet during the first moments of the TTI, would block the possibility of other packets with higher priority that may arrive during the TTI to be transmitted in the next TTI. At $t = 3/4$, another forwarding opportunity occurs. In this case, however, a smaller packet with higher priority arrived into the SDAP between $t = 1/2$ and $t = 3/4$. Since *paced_bytes* increases with the elapsed time (i.e., 1.33×0.75 *TTI* $\times$ 1000 *bytes*/*TTI* + 1500/7 *bytes* = 1211.5 bytes), the next packet to submit is small (i.e., 200 bytes), and thus the variable *extra_bytes* (i.e., 200/5 = 40 bytes), and the RLC buffer is not bloated (i.e., *last_acc_bytes* = 600 bytes and *srs_bytes* = 0 bytes), the packet is forwarded. However, the large packet (i.e., the 1500 bytes packet) is not forwarded during this opportunity, as the *paced_bytes* value did not change (i.e., 1211.5 bytes), and is smaller than the sum of the already sent bytes (i.e., *srs_bytes* = 200 bytes), the *extra_bytes* (i.e., 1500/3 = 500 bytes) and the *last_acc_bytes* (i.e., 600 bytes). Lastly, a forwarding opportunity during the last moments of the TTI ($t = TTI^-$) arrives at the e5G-BDP. This time, e5G-BDP forwards the large packet that it refused to submit before, as the sum of the submitted bytes variable (i.e., *srs_bytes* = 200 bytes), the *last_acc_bytes* (i.e., 600 bytes) and the *extra_bytes* (i.e., 1500/3 = 500 bytes) is smaller than the *paced_bytes* (i.e., 1.33×1.0 *TTI* $\times$ 1000 *bytes*/*TTI* + 1500/7 *bytes* = 1544 bytes). In this manner, e5G-BDP prioritizes utilizing full bandwidth (i.e., it submits more bytes than the calculated bandwidth) at the possible cost of generating some sojourn time. In Fig. 3, if we suppose that the next bandwidth will be equal to the calculated one, the packet forwarded at $t = TTI^-$ will be segmented and some bytes of it shall block the path of future packets. However, through this pacing mechanism, the buffer starvation is avoided, while the sojourn time of packets with different QFIs that share the RLC buffer is reduced.

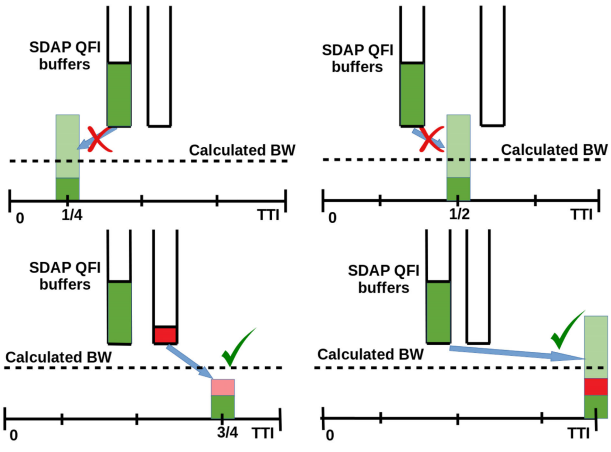

FIGURE 3. e5G-BDP pacing mechanism outcome.

e5G-BDP was explicitly designed to maintain 5G's sublayers decoupled since coupling impedes the 5G foreseen functional split [43], is prone to bugs [44], and is a undesirable software architecture feature overall.

B. ELASTIC QUANTUM PARTITION (EQP)

As described in Section III-B, packet segmentation procedure enables full wireless resource utilization, yet it should be avoided whenever possible as it increases the latency. For example, a strict spectrum based distribution or Fixed Partition (FP) (i.e., fixed number of RBs per RLC buffer every TTI) leads to a plethora of packets waiting at the receivers' RLC buffer, which significantly augments the delay. In Fig. 4, the problem is depicted with four RLC buffers, each containing one packet of 4 RBs, and a bandwidth of 4 RBs/TTI. In the FP approach, one fourth of each packet is sent every TTI. This leads to segmented packets that cannot be forwarded until the 4^{th} TTI, leading to an average delay of 4×4 TTIs / 4 packets $= 4$ TTI/packet. Contrarily, a segmentation/reassembly avoidance distribution would assign all the capacity to one RLC buffer every TTI, so that no segmentation/reassembly happens, and packets can flow to the next sublayer, leading to an average delay of $(1 + 2 + 3 + 4)$ TTI / 4 packets $= 2.5$ TTI/packet.

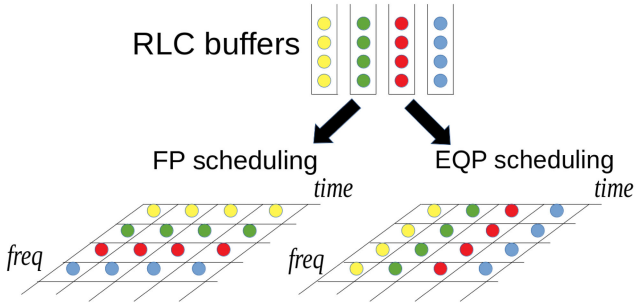

FIGURE 4. FP and EQP RB distribution algorithms.

We mathematically formalize the RLC segmentation/reassembly model as follows. Consider the partially ordered set $(X, \leq)$ composed by N packets at the RLC sublayer $x_1, x_2, \ldots, x_N$, where $N \in \mathbb{N}$. Since packets are egressed in a FIFO order from every RLC queue, there exists a precedence constraint (i.e., to egress a particular packet all the packets that arrived before it have to be first dequeued), and therefore, the precedence relation (i, j) exists in X if item i can only be pulled once j has been pulled. We represent the radio channel capacity with $C \in \mathbb{N}$, and the set R composed of $r_1, r_2, \ldots, r_N$ contains the ceil size in RBs of the RLC SDU packets. These values need to be recalculated every TTI due to the radio link channel dynamicity. As an example, and without loss of generality, let us assume that a RB can transport up to 88 bytes (i.e., the approximate number of bytes per RB with 28 MCS [45]), and that a queue contains two packets of 1500 and 500 bytes, in that precise order. For transmitting the first packet, $\lceil 1500/88 \rceil = 18$ RBs are needed, while for transmitting the first and the second packet, $\lceil (1500 + 500)/88 \rceil = 23$ RBs are required. Hence, adding the second packet, has a total contribution in the number of RBs of $23 - 18 = 5$ instead of $\lceil 500/88 \rceil = 6$. Therefore, that queue would contain a 18 RB packet followed by a 5 RB packet. The objective function to avoid the segmentation/reassembly procedure given the 5G model is to maximize the number of packets pulled from the RLC buffers, given a capacity C during a TTI, which mathematically can be expressed as:

$$max \left(\sum_{i=1}^{N} x_i \right) \tag{1}$$

$$\text{s.t.} \sum_{i=1}^{N} x_i r_i \leq C \tag{2}$$

$$x_i \in \{0, 1\}, \forall i \in N \tag{3}$$

$$x_i \leq x_j, \forall (i, j) \in X \tag{4}$$

The objective function is (1), where the number of packets transmitted is to be maximized. Thus, x can be either 0 (i.e., not selected) or 1 (i.e., selected) according to constraint (3). Constraint (2) models the requirement that the sum of the RBs, where r represents the amount of RBs of a packet, cannot surpass the capacity C. Lastly, (4) models the precedence constraint. If a precedence constraint exists between two packets (i, j) (i.e., they belong to the same queue and the packet x_j arrived before packet x_i), the packet x_i can only be selected (i.e., get the value 1) if the packet x_j has already been selected (i.e., has already a 1 value). This segmentation avoidance algorithm can be reduced to a well-known problem: the *precedence constrained knapsack problem* (PCKP),[3] which is also known as the open pit mining problem. Even though the PCKP is known to be $\mathcal{NP}$-complete [47], there exists a solution that runs in $\mathcal{O}(NC)$. In 5G, the capacity C is a modest number that depends on the channel bandwidth (e.g., in 5G the maximum number of RBs is limited to 275 [42]). This fact mitigates the effect of C in the scheduling algorithm, as no more than the equivalent to 275 RBs per queue have to be considered, and Ethernet packets will be mostly composed by packets larger than one RB.[4] However, N depends on the number of connected UEs, which in commercial base stations can be of the order of 10^4 [48], which should be multiplied by the maximum number of DRBs (i.e., 30 [19]) per UE. The PCKP applied to the RB distribution, maximizes the number of non-segmented packets and, therefore, the delay suffered by segmentation is minimized. However, such scheduling algorithm does not include any fairness. A rogue flow that transmits smaller packets than its competitors would monopolize the access to the resources if we only cared about the objective function (1). Fortunately, the problem of the fairness in a packet based network has

[3]The Weighted Completion Time and Chains [46] algorithm which achieves the optimum scheduling for precedence constrained jobs, does not output the optimum permutation as the time is discrete rather than continuous (i.e., it can only be a multiple of a TTI, but not a fraction of it).

[4]Since the *RB* set contains the ceil value in RBs of the packets, adding a new small packet could result in a 0 size RB packet. This has no practical relevance as (i) no infinite number of packets with size zero can be accumulated as eventually a new RB will be needed and, (ii) packets are usually sent with at least tens of bytes in information to mitigate the header overhead inserted by the transport layer.

been widely studied in the last decades [49], [50]. In [49], the Deficit Round Robin scheduler is presented, where the quantum is used to represent the total bandwidth fraction of each queue in bytes. At every egress opportunity, the corresponding quantum value is added to a state variable that is maintained per queue. If the following packet's size is smaller than the accumulated quantum, the packet is forwarded and the quantum value is reduced according to the packet's size. This procedure is repeated until either the queue is empty or the following packet's size exceeds the quantum value. In this manner, a past bandwidth deficit is fixed in future egress opportunities, achieving fairness. However, in a typical spectrum sharing scenario, the percentage of the total RBs is agreed between different parties through a Service Level Agreement (SLA), and such percentage of RBs is respected within a window time, similarly to the guarantees offered by QFI in 5G (e.g., GBR) [7]. Henceforth, a dichotomy between the short and long term objectives is explicitly presented. On the one hand, in the short term we want to reduce the segmentation that causes delay. On the other hand, inside the agreed window time, we want to respect the percentage of RBs between different queues.

Aiming to bring fairness to our algorithm, we introduced a variable named quantum q. Next, we assume a slicing scenario with a SLA between different slices (i.e., an amount of RBs per slice every TTI). Moreover, an analogous scenario arises whenever a UE has different active DRBs, and thus, our proposed algorithm is also valid in such scenarios. The quantum is increased with the theoretical amount of RBs corresponding to the slice during that TTI (e.g., 12 RBs if there are 24 available RBs and the resource share assigned is 50% of the total RBs). Our algorithm lets slices lend (i.e., quantum to increase) or borrow (i.e., quantum to decrease) RBs within a limit. Consequently, a slice can borrow more RBs than planned in a TTI to avoid packet segmentation and, thus, reduce the delay. However, to avoid unfairness, a limit to the amount of quantum is applied. Such quantum limit depends on the deviation from the agreed percentage of RBs inside the window time (e.g., in a time window of 1 second, with a 1 *ms* TTI, a SLA of 12 RBs and a quantum limit of 100 RBs, a maximum deviation of $max\_possible\_RB\_deviation/total\_RBs = (100 - (-100))/(12 \text{ RBs} \times 1000 \text{ TTIs/sec}) = 1.6\%$ is expected). If the transmission of the following packet decreases the quantum beyond the minimum limit, the access to new RBs is denied. The sum of the resource quantums of all the slices is zero $\sum_{i=1}^{n} q_i = 0$ since the resources lent by a slice are borrowed by another, resulting in a zero sum operation. However, a greedy approach leads to slices working at the quantum limit under certain traffic patterns, and thus, when low-latency packets arrive, an already indebted slice would lack the capability of borrowing enough resources to transmit the low-latency packets rapidly. In the last years, different buffer management policies for packet switches [51] using the competitive analysis [52] have flourished. The competitive analysis measures the optimality of an online algorithm (i.e., an algorithm that takes decisions without knowing the entire input sequence from the beginning) against the optimal offline algorithm (i.e., an algorithm that knows the entire input sequence before starting) performance for any input packet sequence. Even though the cellular network segmentation/reassembly problem cannot be reduced to a known scheduling buffer model due to its complexity (e.g., different packet sizes or dynamic radio link capacity), some outputs can be applied to it. In [53], five different buffer management policies are presented for packets that can have two values (i.e., high or low). The most successful policy (i.e., Dynamic Flexible Partition) accepts low value packets according to the amount of enqueued low value packets and it's free slots through an exponential function. The reasoning being as follows. Since in an online algorithm, the future packet sequence is unknown, the resources for packets that do not contribute to the total reward significantly should be used cautiously, as the algorithm may need the resources in the near future for more profitable packets. We use an analogous approach to discourage an already indebted slice from acquiring more resources. The size of the packets are multiplied by an exponential function that depends on the borrowed RBs (i.e., the quantum value). In this manner, acquiring a larger amount of quantum (i.e., acquiring a new debt) is discouraged when the already borrowed quantum approaches the limit, as the packets seem larger. With this mechanism, the greedy effect of selecting the packets from an indebted slice with small size packets is limited, achieving the objective of reducing the packet segmentation effect while maintaining the fairness in the SLA within a small deviation. To achieve such objectives, we propose the *Elastic Quantum Partition (EQP)* presented in Algorithm 4.

EQP first assigns the corresponding quantum to every slice according to their SLA and the available RBs for the current TTI (i.e., in a 50% slice where 24 RBs can be allocated, $\lfloor 24 \times 0.5 \rfloor = 12$ RBs). It then converts the current queues with packets in bytes, into queues with RBs considering the cellular network characteristics (e.g., the MCS), as well as the quantum, through the *generate_rbs* function. Packets belonging to indebted slices (i.e., negative quantum) are considered larger (i.e., their number of bytes are multiplied by an exponential function that depends on the fraction *quantum borrowed*/*quantum limit*). In line 3, Algorithm 4 calls the segmentation avoidance algorithm. It returns a permutation of packets considering the slices' quantum that does not trigger the segmentation/reassembly procedure. Next, EQP assigns the RBs to the permutation returned by the *seg_avoid* function and updates the remaining *total_rbs*, as well as the slices' quantum. However, some RBs may still be unallocated (i.e., *total_rbs* may not be 0). Therefore, the slices are sorted in quantum descending order and the next UE (i.e., the Radio Network Temporary Identifier (RNTI) is a temporal identifier for a UE) selected. The empty queues, as well as the queues that will get drained in the next TTI (i.e., the already assigned RBs in lines 4 – 8 will empty them),

are excluded from the sorting. If there are still unallocated RBs and the slice quantum is not indebted above the 75% of the quantum maximum limit, the RBs are distributed considering 5G specificities (i.e., using RBG). Such mechanism objective is twofold. On the one hand, it avoids squandering RBs. For example, in a two slice scenario, if the buffers of the first slice are empty while the second slice buffers contain packets, the second slice can use the RBs from the first slice to minimize the unused RBs, and thus, augment the total throughput. On the other hand, it limits the RBs assigned to a slice that does not achieve forwarding a full packet. In this way, slices maintain a quantum buffer of around 25% of the total quantum limit, in case that these RBs are needed during the next TTIs. This is important in an online scenario, such as 5G, where the traffic patterns cannot be foreseen, and a slice may use the quantum in a more rewarding manner (i.e., being able to forward a larger amount of packets). Lastly, the remaining RBs are assigned to the slices with the highest positive quantum (i.e., line 19), even if the buffers of these slices are empty. Such a decision sacrifices some throughput in favor of fairness, since EQP interprets the lack of more packets in a slice as a desired symptom (e.g., the application may not have more information to transmit) and abides by the SLA.

Algorithm 4: *Elastic Quantum Partition* (EQP). It Maps the Free RBs to Non Empty RLC Buffers

Input: Total number of RBs to schedule and active RNTI (*total_rbs* and *active_rntis*).

1: *assign_quantum_slices*();
2: $idx\_q = generate\_rbs(total\_rbs, active\_rntis)$;
3: $out\_arr = seg\_avoid(idx\_q, 0, total\_rbs, out\_arr)$; // Alg. 5
4: **for all** $pkt, rnti, slice \in out\_arr$ **do**
5: *assign_rbs*(*rnti*, *pkt*);
6: *reduce_slice_quantum*(*slice*, *pkt*);
7: $total\_rbs -= pkt$;
8: **end for**
9: *sort_slices*();
10: $rnti = next\_rnti()$
11: **while** $total\_rbs > 0 \wedge slice\_quantum(rnti) > -0.75 \times limit\_quantum$ **do**
12: *assign_rbs*(*rnti*, *min_rbg*);
13: *reduce_slice_quantum*(*slice*, *min_rbg*);
14: $total\_rbs -= min\_rbg$;
15: *sort_slices*();
16: $rnti = next\_rnti()$
17: **end while**
18: **if** $total\_rbs > 0$ **then**
19: *map_last_RB_slices*()
20: **end if**

Algorithm 5 presents the segmentation avoidance algorithm. It consists in a recursive algorithm where

Algorithm 5: EQP *seg_avoid*. It Selects a Permutation of Packets According to Their Size and the Slice Quantum That Does Not Cause Segmentation

Input: Queue index, packet index, capacity and permutation of selected packets (*idx_q*, *idx_pkt*, *capacity* and *out_arr*)
Output: Permutation of selected packets (*out_arr*)

1: **if** $capacity == 0$ **then**
2: **return** *out_arr*;
3: **end if**
4: **if** $idx\_q > max\_idx\_q$ **then**
5: **return** *out_arr*;
6: **end if**
7: **if** $is\_memoized(idx\_q, idx\_pkt, capacity)$ **then**
8: **return** $memoized(idx\_q, idx\_pkt, capacity)$;
9: **end if**
10: $p\_size = pkt\_size(idx\_q, idx\_pkt)$;
11: $t\_capacity = adjust\_rbgs(capacity, out\_arr, p\_size)$;
12: **if** $t\_capacity < 0$ **then**
13: $idx\_q = next\_idx\_q(idx\_q)$;
14: $idx\_pkt = 0$;
15: **return** $seg\_avoidance(idx\_q, idx\_pkt, capacity, out\_arr)$
16: **end if**
17: $idx\_pkt += 1$;
18: **if** $idx\_pkt == max\_idx\_pkt(idx\_q)$ **then**
19: $idx\_q = next\_idx\_q(idx\_q)$;
20: $idx\_pkt = 0$;
21: **end if**
22: $act\_pkt = \{idx\_q, idx\_pkt\}$;
23: $take = seg\_avoid(idx\_q, idx\_pkt, t\_capacity, out\_arr + act\_pkt)$;
24: $dont\_take = seg\_avoid(next\_idx\_q(idx\_q), 0, capacity, out\_arr)$;
25: $out\_arr = dont\_take$;
26: **if** $len(take) > len(dont\_take)$ **then**
27: $out\_arr = take$;
28: **else if** $len(take) == len(dont\_take) \wedge quantum(take) > quantum(dont\_take)$ **then**
29: $out\_arr = take$;
30: **end if**
31: $memoization(idx\_q, idx\_pkt, capacity, out\_arr)$;
32: **return** *out_arr* ;

a memoization[5] (i.e., at lines 7, 8 and 31) in the packet position (i.e., *idx_q* and *idx_pkt*), and the *capacity* is implemented with the goal of obtaining a $\mathcal{O}(NC)$ algorithmic complexity rather than $\mathcal{O}(2^N)$. Through memoization, repetitions of already calculated permutations in the packet index, queue index and capacity are avoided.

[5]*Memoization* is not a typo from the word *memorization* but rather an optimization technique.

Due to the recursive nature of Algorithm 5, we will start presenting it from the middle rather than from the beginning. The core of the segmentation avoidance algorithm resides at lines 23 and 24. The algorithm either selects the current packet (i.e., adding the current packet *act_pkt* in the possible permutation) and reduces the remaining capacity accordingly, or jumps to the next queue maintaining the current capacity, generating all the valid permutations along the way. Once the stack unwinds, two possible permutations are available (i.e., the permutation of selecting the current packet (i.e., *take*) and reduce the capacity (line 23) or ignore it and jump to the next queue maintaining the available capacity (i.e., *dont_take*) (line 24)). At line 26, the permutation with the largest amount of packets is selected. If lengths are equal, the solution with the largest sum of the quantum packets is selected at line 28 (i.e., every packet belongs to a slice with a quantum associated, so the larger sum of the quantum indicates the permutation of packets that reside at slices that lent more RBs). Such measurement balances the quantum values of the slices and achieves higher fairness in the case where the same amount of packets can be pulled. The algorithm lastly saves the obtained result (i.e., *out_arr*) according to the *idx_q*, *idx_pkt* and *capacity* at line 31 before returning.

As input, Algorithm 5 receives the queue index (i.e., *idx_q*), the packet index (i.e., *idx_pkt*), the remaining capacity (i.e., *capacity*) and a copy of the ongoing selected array permutation (i.e., *out_arr*). The termination conditions of the recursive Algorithm 5 are coded in the first lines (i.e., lines 1 – 6). At line 1, the capacity is checked, and if zero (i.e., no more RBs available), the current packets permutation (i.e., *out_arr*) is returned. At line 4 the end of the queues (i.e., no more queues to consider) are checked, and if the end is reached, returned. At line 7, whether the packet permutation has already been resolved is checked, thus avoiding unnecessary work and reducing the algorithm complexity. At line 10, the packet size in RBs according to its queue index (i.e., *idx_q*) and packet index (i.e., *idx_pkt*) is acquired (i.e., *p_size*). At line 11, the remaining capacity is obtained (i.e., *t_capacity*) if the current packet is selected in the permutation (i.e., packet at queue index *idx_q* and packet index *idx_pkt*), with 5G's RBG specificity adjusted. If after considering the 5G RBG distribution, the capacity drops below zero, this combination is discarded and the algorithm recursively calls itself at line 15, after updating the queue and packet index. If the current packet is the last one in the queue, the queue and the packed indexes are updated (i.e., *idx_q* and *idx_pkt* at lines 19 and 20).

In summary, Algorithm 5 generates a packet permutation where the number of packets to forward without generating segmentation is maximized (i.e., line 26). In case that two permutations would forward the same number of packets, Algorithm 5 selects the permutation with the largest quantum (i.e., line 28), and thus, it reduces the unfairness between slices, as the slices with larger quantum lent more RBs in the past. Lastly, Algorithm 5 is aware of the RBG distribution and adjusts the remaining capacity accordingly through the function *adjust_rbgs*.

V. EVALUATION FRAMEWORK

In this section a complete overview of the evaluation conditions, the testbed configuration and the measurement methodologies are exposed.

A. SOFTWARE

To evaluate and validate the proposed e5G-BDP and EQP algorithms, a real cellular network testbed using OpenAir-Interface (OAI) [45] was built. For this, we enhanced the OAI implementation with the i) SDAP sublayer with QFI queues and ii) the scheduler proposed in Section II. Downlink traffic was generated to validate the results, and analogous outcomes are expected in uplink procedure due to 5G stack's symmetry. We configured OAI's channel bandwidth to 5 MHz (i.e., 25 RBs). The TTI in OAI is 1 *ms*. The default bearer was mapped to a RLC-UM bearer, as it was OAI's default option. The SDAP scheduler iteratively asks e5G-BDP *limit_reached* function (i.e., Algorithm 1) whether it should forward a packet or maintain it. However, OAI is a soft real time system where an excessive burden to the CPU leads to de-synchronize the UE and the eNodeB. Therefore, a 200 μs sleep between the iterative calls was applied, which heuristically proved correct for the hardware tested.

B. HARDWARE

As seen in Fig. 5, a B-200 Ettus USRP connected to a computer with an Intel(R) Core(TM) i9-9980HK CPU @ 2.40 GHz processor with the Linux 5.3.0-51 low-latency kernel was deployed at the eNodeB side. For the first UE (i.e., UE0), a Commercial off-the-shelf (COTS) Huawei E3372 LTE USB stick connected to a Raspberry Pi 4 Model B was used. For the second UE (i.e., UE1), another Huawei E3372 LTE USB stick connected to a computer with an AMD FX(TM) 8120 CPU @ 1.40 GHz processor with the Linux 4.4.0-141-generic kernel was used.

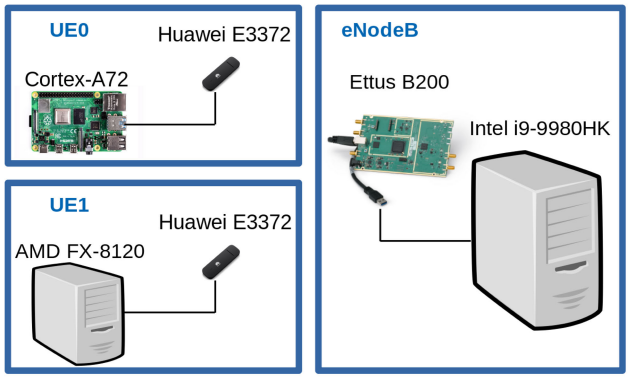

FIGURE 5. Evaluation testbed.

C. RADIO LINK CHANNEL CAPACITY

The radio link plays a central role establishing the channel capacity. To realistically emulate the channel capacity, and thus, validate our proposals as reliably as possible, we converted the CQI data gathered from two major Irish operators [54] into MCS sequences according to OAI's CQI to MCS conversion function. In [54], LTE UE statistics for five different mobility patterns (i.e., static, pedestrian, car, tram and train), at a granularity of one sample per second, are provided. We selected two mobility patterns (i.e., pedestrian and train), thus, exposing our solutions to diverse, and yet realistic patterns. Every second, we set the MCS according to the data provided by the mobility patterns file, instead of the MCS that would had been assigned in accordance with the CQI provided by the UE (i.e., in our testbed conditions the UE always reports the maximum possible CQI), and therefore, we realistically emulated the dynamic radio link channel capacity.

D. BUFFERBLOAT SOLUTIONS CONFIGURATION

We implemented the six solutions shown in Table 1 to compare and validate our proposed e5G-BDP. The Vanilla case represents the default OAI implementation. It does not have any cross-layer communication mechanism or pacer, is knob-less and does not drop packets. For the evaluation of CoDel, we substituted the default RLC FIFO buffer with a CoDel queue according to the values described in [11] (i.e., interval value set to 100 *ms* and target value set to 5 *ms*). CoDel, contrarily to the Vanilla implementation, drops packets to inform the sender that excessive buffering is happening. For the BBR [27] evaluation, we used the default TCP BBR version from the Linux kernel 5.3.0-62-lowlatency. As explained in Section III, BBR calculates the BDP and transmits the packets accordingly through a pacing mechanism. Lastly, the three algorithms that permit a rapid low-latency packet delivery from within the cellular network due to their cross-layer communication capability were evaluated: DynRLC, DRQL and e5G-BDP. DynRLC measures the sojourn time suffered by the SDUs at the RLC, and consequently adjusts the maximum number of allowed SDUs in the RLC buffer. The sojourn time is captured for every packet that egresses the RLC buffer, and depending on whether the sojourn packet time is bigger or smaller than a target delay, the number of SDUs permitted at the RLC is increased or reduced accordingly. An α parameter marks the increase or reduction rate. The new maximum number of allowed SDUs is updated every interval of time [22], which we set to 10 *ms* (i.e., a radio frame duration in LTE), in contrast to the 200 *ms* mentioned in [22]. We set the target delay to 3 *ms* instead of the minimum of 30 ms reported by [22], as a smaller value was heuristically found not to achieve full bandwidth, and a larger value just adds unnecessary sojourn time. We set the maximum number of packets to 10, the minimum to 1 and the α value to 0.05. DynRLC lacks any pacing capability as elapsed time is not considered while forwarding the packets. DRQL [24], on the contrary, is based on the number of bytes that remain at the RLC buffer after a TTI, and no parameters need to be adjusted as DRQL tries to maintain the equivalent of one MTU in the RLC buffer after the TTI. Packets are also forwarded in a bursty manner, as DRQL also lacks a pacing mechanism. Lastly, e5G-BDP measures the bandwidth directly through the amount of bytes that were pulled from the RLC buffer in the last TTI. It also owns a pacing mechanism to augment the probabilities of delivering a packet within a TTI, and is a knob-less solution as no parameters need to be configured. We did not use any slicing to evaluate the bufferbloat as it would not report any advantage over the simple scenario with 1 UE.

TABLE 1. Bufferbloat algorithms tested.

	Van.	CoDel	BBR	e5G-BDP	Dyn.	DRQL
X-layer	No	No	No	Yes	Yes	Yes
Pacing	No	No	Yes	Yes	No	No
Knob-less	Yes	Yes	Yes	Yes	No	Yes
Drop pkts	No	Yes	No	No	No	No

E. EQP CONFIGURATION

To test the EQP we created two slices through FlexRAN [55], each one containing a single UE. We evaluated different SLAs for slices with 75%, 50% and 25% of the total floor number of RBs (e.g., for 25 RBs this is translated to $\lfloor * \rfloor(25 * 0.75) = 18$, $\lfloor * \rfloor(25 * 0.50) = 12$ and $\lfloor * \rfloor(25 * 0.25) = 6$ RBs per TTI.). In this manner, we evaluated EQP for slices with large, as well as small amounts of RBs, emulating different, yet realistic, slicing scenarios. OAI's default slice SLA was preserved, where the slices are generated according to a percentage of the total data RBs excluding the retransmissions (i.e., due to HARQ/NACK mechanism), as well as the RBs assigned to transmit control information. We compared our proposed EQP solution against the default resource scheduling in OAI for slicing. We named such distribution FP as it corresponds to a fixed partition, which is independent of the size of the packets. Since the bufferbloat effect introduces a delay several orders of magnitude larger than the segmentation/reassembly procedure, we evaluated EQP in conjunction with e5G-BDP due to its superior results when compared with its direct competitors, as proved by this paper. We also tested the effect of the quantum limit through five different values (i.e., 25, 50, 100, 250 and 500 RBs) in a 25% slice with 8 VoIP flows, as it represents our most challenging scenario for EQP. We set the exponential function of the function *generate_rbs* to $5 \times (quantum\ borrowed / quantum\ limit)^5$ to discourage considering packets from indebted slices for scheduling, and a quantum value limit of 100 RBs was set as the default value as at demonstrated by this paper, it avoids a possible EQP saturation for the scenarios evaluated.

F. TRAFFIC GENERATION

To model realistic low-latency flows, the Isochronous Round-Trip Tester (*irtt*) [56] was used, where a G.711 VoIP conversation is emulated through UDP data frames

of 172 bytes with an interval of 20 *ms* [31], resulting in a bandwidth consumption of 64 Kbps. Conversely, we used the *iperf3* tool in reverse mode to generate a bulky TCP flow that models a service that demands bandwidth (e.g., a mobile application update), with a maximum segment size (i.e., MTU minus 40 IP header bytes) of 1460 bytes. We chose TCP as the competing flow as most of the Internet traffic is forwarded through HTTP/2, which relies on TCP [57] as its transport layer protocol. In this manner, we modelled two different flows (i.e., one bulky and one with low-latency requirements) that try to access limited resources and share the RLC buffer. Both flows were always segregated at the SDAP sublayer according to their 5-tuple, and the UDP flow's packets were placed in a higher priority queue, imitating the behavior of two flows with distinct QFI. TCP Cubic [8] was used for all the bandwidth driven flows, except when BBR [27] is explicitly mentioned, as it is the most deployed TCP congestion control algorithm. To achieve TCP's steady state, avoid its slow start, and generate the bufferbloat, the bulky traffic started 5 seconds before the VoIP flow. Therefore, the VoIP flow avoided possible outlier results generated at the beginning of the transmission and the repeatability of the results was improved. Every VoIP lasted for 60 seconds, which represents 60000 TTIs (i.e., 60000 *ms* x 1 TTI/*ms*) in which 3000 UDP packets were sent. Every test was repeated 5 times (i.e., 300000 TTIs) to correctly verify the results and minimize the possible outliers' effect. Scalability was evaluated with 1, 2, 4 and 8 VoIP flows in parallel, thus modeling a group phone call conference scenario with background TCP traffic.

G. MEASUREMENT METHODOLOGIES

To report the delay, a timestamp was added to all the packets once they enter the SDAP sublayer in the case of DynRLC, e5G-BDP and DRQL, or the RLC sublayer in the case of Vanilla, BBR and CoDel. The elapsed time was measured in the VoIP packets once they were forwarded to the MAC sublayer, and thus, the HARQ/NACK delay was not measured in our setup. By doing so, the delay that occurs in the SDAP/RLC sublayers was isolated, and thus, the effectiveness of the proposed solutions could be verified minimizing the effect of other possible phenomena. The Round Trip Time (RTT) was reported through the *irtt* tool, which generated the VoIP flows. The achieved bandwidth was measured through the *iperf* 3 tool (i.e., counting the TCP packets that arrived to the UE), as well as counting the unused transmission opportunities that happened every TTI at the eNodeB (i.e., unused RBs). In this manner, the throughput was verified from two different perspectives. A UDP packet was sent from the UE before the beginning of the emulated VoIP phone call to start reading the MCS profile, when different scenarios (i.e., train and pedestrian) were involved.

VI. EVALUATION RESULTS

In this section, we first evaluate the algorithms that address the bufferbloat (i.e., Vanilla, CoDel, BBR, e5G-BDP, DynRLC and DRQL). To verify their suitability they are tested in a static scenario (i.e., 28 MCS, one VoIP flow and one bulky flow) and a dynamic scenario (i.e, using the train and pedestrian reported MCS, one VoIP flow and one bulky flow). Their scalability is tested with 1, 2, 4 and 8 VoIP flows in parallel, emulating a group phone call conference scenario along a bulky traffic flow that tries to monopolize the access to the RBs.

Following, we show the effect of RLC's segmentation/reassembly procedure and evaluate EQP. To this end, we first show the results in a slicing scenario with different RB percentage shares (i.e., 25%-75% and 50%-50%) and a VoIP and a bulky flow with a fixed 28 MCS. Next, the results obtained from a scalability test in a 25% RB slice with 1, 2, 4, and 8 VoIP flows in parallel along a bulky flow are displayed. Following the effect of the quantum limit in a 25% slice with 8 VoIP flows in parallel is evaluated. We finish this section showing the effect of EQP on two UEs with 50% of the total RBs each, that belong to two different mobility patterns MCS (i.e., pedestrian and train).

A. RLC BUFFERBLOAT

1) STATIC MCS SCENARIO

3GPP standard does not include any mechanism capable of addressing the bufferbloat problem in current cellular networks. Therefore, 5G's stack is susceptible to experience large delays, as it is equipped with large buffers, the RAN is commonly the slowest link of the data path, and packets are usually transmitted through a loss-based TCP congestion control algorithm (e.g., TCP Cubic). Thus, packets with low-latency requirements that share the RLC buffer with bulky flows suffer from large buffer depletion times. Such an effect can be clearly seen in Fig. 6, where the correlation between the VoIP packet's delay and the RLC queue size can be clearly observed in an OAI Vanilla deployment, when one low-latency flow shares the RLC buffer with a bulky flow. The delay suffered by low-latency flows due to bloated buffers generated by bulky flows in Vanilla deployments, can reach the order of seconds as reported by [9] and shown in Fig. 6.

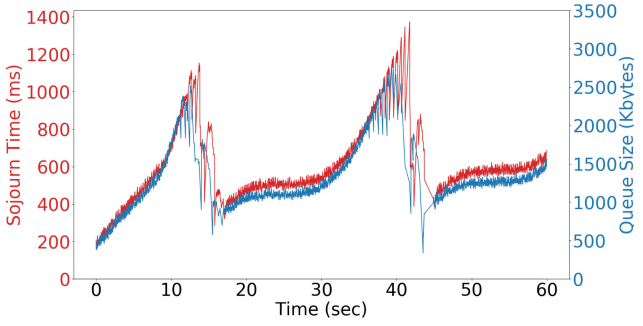

FIGURE 6. Delay suffered by a VoIP flow when sharing the RLC buffer with a bulky flow and the RLC buffer occupancy in a Vanilla OAI deployment.

The RLC buffer should contain just enough bytes to satisfy the MAC sublayer demands. Any additional byte just augments the delay [24], as newly arrived data packets suffer the depletion time of the previously enqueued packets. On the other hand, if the RLC cannot satisfy MAC sublayer's demands, a transmission opportunity is squandered. Fulfilling both objectives is very challenging in the 5G environment due to the dynamic nature of the radio link channel. In Fig. 7, the RLC buffer occupancy for the six different methods from Table 1 can be observed, where a bulky flow shares the RLC buffer with a VoIP flow during 60 seconds. Note the change of scale between the different solutions, where CoDel and BBR reduce the amount of bytes at the RLC queue by at least 5 times when compared with Vanilla. However, CoDel's mechanism for discarding packets results in transmission opportunity losses (i.e., not containing enough bytes at the RLC buffer when a MAC notification arrives). On the other hand, BBR periodically drains the bottleneck queue after 10 seconds during 200 *ms* if the measured RTT does not decrease in that period [27]. This leads to a small total bandwidth utilization reduction, as the RLC buffer does not contain enough bytes to satisfy the MAC sublayer's TBS demands during that 200 *ms* periods. This effect can be clearly observed in Fig. 7, as the BBR RLC buffer is drained several times during the emulated 60 seconds VoIP call. Moreover, Vanilla, CoDel and BBR create a large queue that impedes a fast packet delivery. Conversely, the algorithms that are deployed directly in the cellular network stack (i.e., e5G-BDP, DynRLC and DRQL), estimate more accurately the data link capacity, forwarding enough bytes to feed the MAC requests every TTI, while maintaining the rest segregated at the higher sublayer queues (i.e., SDAP). Thus, a low-latency flow, where e5G-BDP, DynRLC or DRQL is implemented, sharing the bottleneck buffer (i.e., RLC buffer in 5G) with bulky traffic flows considerably avoids large queuing sojourn times, as the amount of bytes is meticulously maintained low.

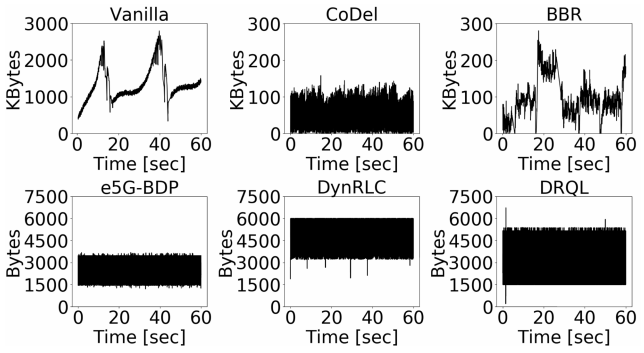

FIGURE 7. RLC buffer size for Vanilla, CoDel, BBR, e5G-BDP, DynRLC and DRQL with MCS 28.

As observed in Fig. 7, e5G-BDP is able to maintain the RLC buffer with fewer bytes than DynRLC or DRQL. Under a MCS index of 28 and 25 RBs in OAI, the MAC scheduler pulls between 2289 and 1569 bytes, depending on whether the subframe contains control and data information or just data. Hence, the queues shown at Fig. 7 should never contain less than 2289 bytes to avoid wasting any transmission opportunity, while any additional byte just increments the delay. In our evaluation, the bulky flow packets are 1500 bytes long, while the packets from the low-latency flow are 200 bytes long (i.e., 172 bytes of data, 20 bytes of IPv4 header and 8 bytes of UDP header). Consequently, precisely occupying the RLC queue with 2289 bytes may not be possible in many TTIs. Hence the problem between the packet sizes and the TBS is explicitly shown, as the TB transmits information in bits while the RLC buffer stores packets. It is not possible to pass bits from the SDAP sublayer to the RLC sublayer, and thus, to assure full throughput, the RLC buffer should always contain more than 2289 bytes, which increases the sojourn time.

In Fig. 8 the same conclusions from the RTT of the VoIP flow's perspective reported by *irtt* are explicitly drawn. The RTT is composed by the cellular stack delay (i.e., uplink and downlink procedure), the cellular protocol delay (e.g., the uplink Scheduling Request procedure), the packet routing and various buffering delays (e.g., Network Interface Controller (NIC) buffers). However, the importance of segregation in the cellular downlink procedure is clearly shown in Fig. 8, where the packets sent through e5G-BDP, DynRLC and DRQL surpass the alternatives that do not segregate the packets (i.e., Vanilla, CoDel and BBR). e5G-BDP also surpasses DynRLC and DRQL as observed in Fig. 8. It can be also concluded that in our cellular network testbed a minimum of 20 ms RTT delay exists.

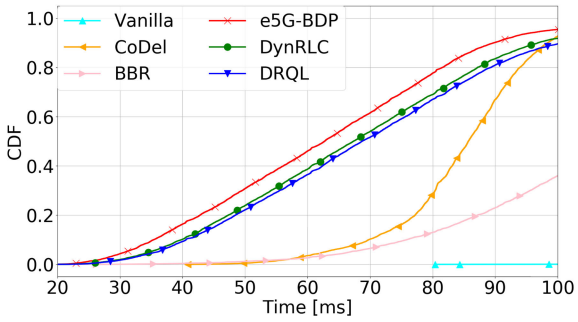

FIGURE 8. Cumulative Distribution Function (CDF) of the VoIP flow's RTT.

Fig. 9, depicts the CDF of the queuing delay in the downlink procedure for the SDAP sublayer, the RLC sublayer and the sum of both. Packets sent with Vanilla, CoDel and BBR algorithms face an order of magnitude larger delay than packets sent with e5G-BDP, DynRLC or DRQL. The time where Vanilla, CoDel and BBR's CDF asymptotically converge into 1.0 are located far after the scope of Fig. 9. Indeed, only 12%, 76% and 91% of the total packets at Vanilla, BBR and CoDel, suffer less than 50 ms of delay. Vanilla, CoDel and BBR lack an SDAP sublayer, and therefore, no graph is depicted in the first row of the Fig. 9. Between e5G-BDP, DRQL and DynRLC, the lack of a pacing mechanism in DRQL and DynRLC

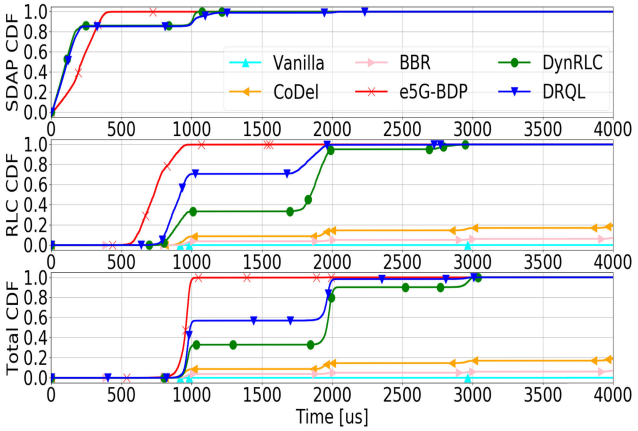

FIGURE 9. CDF of the queuing delay at RLC, SDAP and the sum of both for a 28 MCS.

has notorious effects. Every new TTI, the SDAP scheduler forwards the packets of non-empty queues according to its priority policy (i.e., in our evaluation scenario it schedules low-latency packets first). This effect can be clearly seen in Fig. 9 at the RLC buffer, where the packets are forwarded from the SDAP sublayer at the beginning of the TTI and have to wait at the RLC buffer until the next transmission opportunity occurs (i.e., most of DynRLC and DRQL packets at RLC wait between (750, 1000) μs, meaning that they were forwarded from the SDAP sublayer in the (0, 250) μs time interval after a TTI). Moreover, if the packets have not been forwarded at the beginning of the TTI, they have to wait at the SDAP sublayer until the next TTI opportunity (i.e., an additional delay of 1000 μs is added at the SDAP sublayer for DynRLC and DRQL). On the other hand, the pacing capabilities of e5G-BDP permit that a newly arrived packet at the SDAP is forwarded to the RLC immediately, avoiding having to wait for the next TTI. Such effect is also appreciable at the RLC queue, where the e5G-BDP permits ingressing packets, due to its pacing capabilities, at any time within a TTI, if the limit has not been reached, considerably reducing the delays at the RLC buffer (i.e., the packets stay mostly at the RLC sublayer between (500, 1000)) μs). This pacing capability allows e5G-BDP to avoid waiting for a new TTI, and therefore, outperforms DRQL and DynRLC as shown in Fig. 9, delivering more than 95% of the packets within a TTI, while only around 50% and 30% of the packets using DRQL and DynRLC are delivered during the first 1000 μs.

Achieving low-latency for the new cellular network is of vital importance. However, low-latency can be trivially achieved if some throughput is sacrificed, simply maintaining the RLC buffer with less bytes than the MAC requested TBS. Therefore, the real challenge is to achieve low-latency while not starving the RLC buffer, and thus, utilizing all the available throughput. To this end, the throughput is quantified from two perspectives. First we measured the results from *iperf* 3 (i.e., the bulky flow) that rely on Layer 4 measurements. The results can be observed in Fig. 10, where Vanilla,

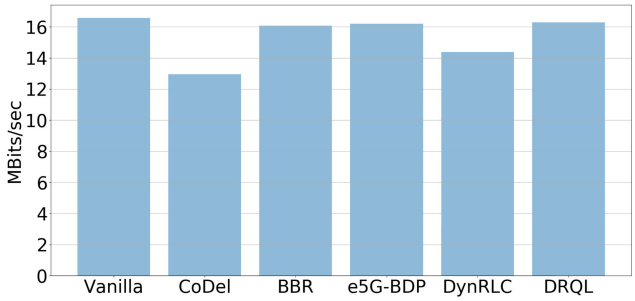

FIGURE 10. Average throughput reported by *iperf* 3 with MCS 28.

CoDel, BBR, e5G-BDP, DynRLC and DRQL reported 16.58, 12.95, 16.05, 16.20, 14.35 and 16.30 MBits/sec. As a second approach, we compared the number of unused RBs during the 60 second VoIP conversation. The Vanilla case transports data information in all the RBs during the 60 seconds and, therefore, achieves the highest possible throughput. BBR, as shown in Fig. 7, drains the buffer to get a new measurement of the bottleneck link path every 10 seconds. This effect contributes to wasting 3.1% of the total RBs during the 60 seconds of VoIP conversation. The mechanism of discarding packets utilized by CoDel results in a 20.92% of transmission opportunity losses. DynRLC and DRQL endure a 10.12% and a 1.3% unused RBs. Lastly, e5G-BDP does not transmit data information in 1.9% of the total available RBs. These results match the outcomes provided by *iperf* 3 as shown in Fig. 10. From this scenario, e5G-BDP clearly outperforms its competitors. While some RBs are squandered (i.e., 1.9%), more than 95% of the low-latency packets are forwarded within a TTI (i.e., 1000 μs) as shown in Fig. 9, while DRQL needs two TTIs and DynRLC needs three TTIs to forward more than 95% of the packets. A summary of the results can be observed in Table 2.

TABLE 2. Performance comparison of static MCS scenario: Percentage of packets with < 1 *ms* queuing delay, the throughput of *iperf* 3 flow in MBits/sec, and the percentage of used RBs.

	Van.	CoDel	BBR	e5G-BDP	Dyn.	DRQL
Delay	0%	9%	4%	**95%**	30%	50%
Throu.	**16.58**	12.95	16.05	16.20	14.35	16.30
RBs	**100%**	79.08%	96.9%	98.1%	89.88%	98.7%

2) SCALABILITY EVALUATIONS

In contemporary cellular networks, different flows with heterogeneous QoS requirements share the RLC buffer. Thus, the presented solutions must scale well when the number of flows increases. To this end, we tested the effect of 1, 2, 4 and 8 VoIP flows in parallel along a bulky flow, emulating a group phone call conference scenario. Note that DynRLC relies on the number of SDUs in the RLC and the delay they suffer to adjust the amount of SDUs at the buffer. However, the MAC sublayer requests bytes from the RLC. This creates a weakness on DynRLC when packets of different sizes are mixed (i.e., DynRLC does not differentiate between a 1500 or

200 bytes SDU), thus wasting transmission opportunities (i.e., not containing enough bytes at the RLC buffer when the MAC requests them) and creating additional delay (i.e., when low-latency packets bursts occurs, they may not all be forwarded to the RLC if they exceed the number of optimal SDUs). Additionally, as observed in Fig. 11, it does not scale well in comparison with e5G-BDP and DRQL that are methods that rely on the amount of bytes rather than the number of SDUs.

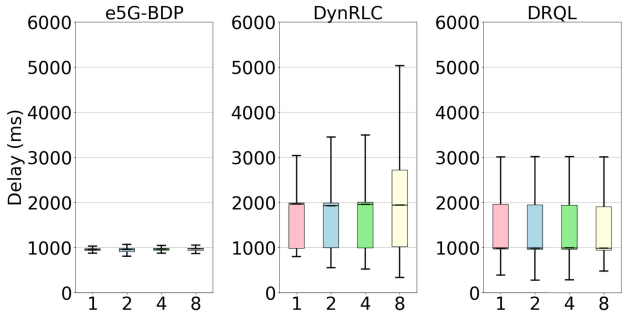

FIGURE 11. Delay faced by VoIP packets with 1,2,4 and 8 flows with 28 MCS.

3) DYNAMIC MCS SCENARIO

We obtained the previously shown results in a controlled static environment with the maximum MCS index (i.e., 28) and nearly no interference. However, in real cellular network deployments the radio link channel capacity abruptly changes [54]. Therefore, we emulated a scenario with dynamic MCS, setting its value according to real LTE traces [54]. We chose a pedestrian scenario, where the MCS differs more slowly, and a train scenario, where the MCS changes more abruptly. In Fig. 12, the MCS values from these two traces and the number of bytes requested by the MAC sublayer during the 60 seconds VoIP conversation can be observed. As seen in Fig. 12, they are highly correlated, and therefore, a MCS reduction directly affects the available bandwidth during the following TTI (i.e., bytes requested/TTI). The sudden MCS variation at the 35th second in the train scenario presents a challenge for the algorithms, as they have to rapidly estimate the data link bandwidth, so that the bufferbloat is prevented, and thus, the delays associated with it. Under such conditions we can observe how the proposed algorithms behave in Fig. 13. A reduction at the 35th second in the amount of bytes at the RLC buffer can be observed for CoDel, BBR, e5G-BDP and DRQL, while no specific change at Vanilla and DynRLC is shown. The dynamic MCS scenario shows DynRLC limitations, as its buffer capacity calculation is performed according to the number of SDUs. Therefore, the amount of lost transmission opportunities is increased. However, e5G-BDP still delivers the low-latency packets faster than DynRLC as it can be observed from Fig. 14, due to the lower sojourn times experienced by the low-latency flows at the SDAP sublayer. No algorithm is capable of fully utilizing the bandwidth and, at the same time, forward the low-latency packets within

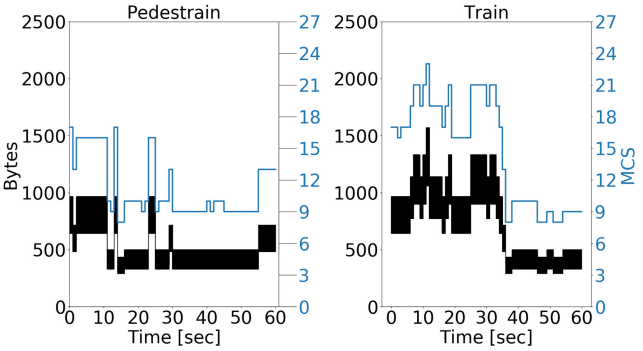

FIGURE 12. Bytes requested by the MAC sublayer and MCS for the train and pedestrian datasets.

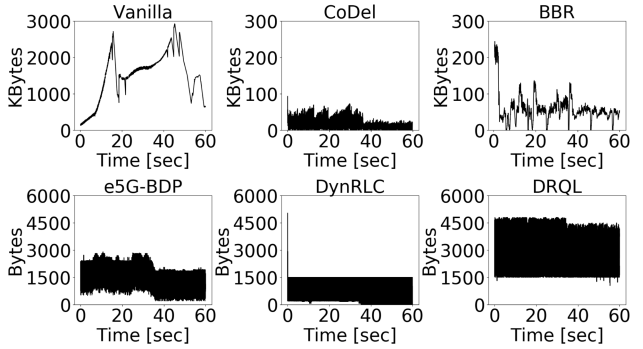

FIGURE 13. RLC buffer size for Vanilla, CoDel, BBR, e5G-BDP, DynRLC and DRQL for the train dataset.

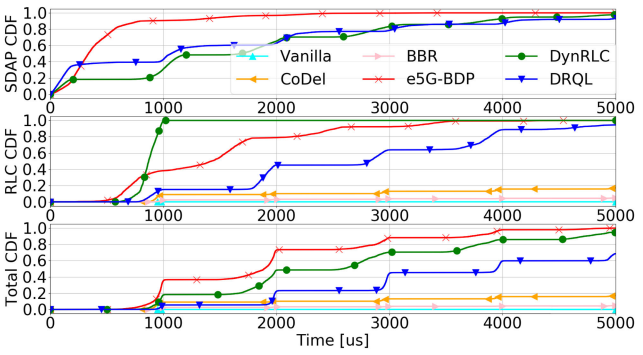

FIGURE 14. CDF of the queuing delay at RLC, SDAP and the sum of both for the train dataset.

a TTI. Such outcome results from the fact that no preemption is allowed from the RLC buffer in our model. Therefore, if a packet of 1500 bytes has already been forwarded from the SDAP to the RLC and just 500 bytes are forwarded from the RLC to the MAC every TTI, a newly arrived low-latency packet will suffer at least the depletion time from the previous packet (i.e., at least 3 TTIs or 3000 μs).

The average throughput for the train scenario reported by *iperf3* can be observed in Fig. 15, with 6.20, 5.20, 6.20, 5.98, 4.30 and 6.0 MBits/sec for Vanilla, CoDel, BBR, e5G-BDP, DynRLC and DRQL.

One of the most important features when analyzing the bufferbloat problem is the bandwidth vs. delay dichotomy.

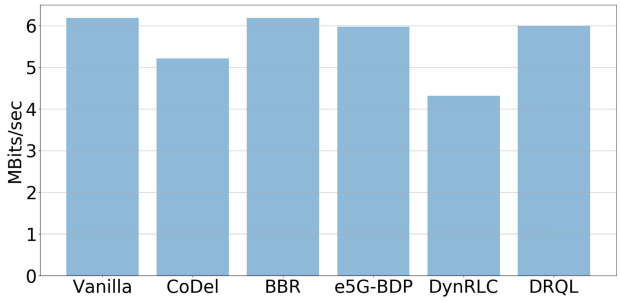

FIGURE 15. Average throughput reported by *iperf3* for train dataset.

It is relatively easy to obtain the lowest possible latency if some bandwidth is squandered and, analogously, it is relatively easy to obtain full bandwidth if large queues (i.e., bufferbloat) can be formed. However, achieving both is very challenging as not only the dynamic cellular network radio link channel effects have to be taken into account, but also TCP's behavior has to be considered. In Fig. 16, the sojourn time from the low-latency flows vs. the normalized amount of used RBs for the train and pedestrian scenarios is depicted. A RB is composed of several bytes and therefore, a small mismatch between the utilized RBs and the real bandwidth may exist. In Fig. 16 the solutions that nearly maintain full bandwidth while preventing the generation of the bufferbloat, and hence large delays, at the RLC buffer are e5G-BDP and DRQL. DynRLC suffers from the previously mentioned effect of calculating the delay according to the number of packets at the RLC, which maintains the queue depleted, albeit squandering transmission opportunities, as also seen in Fig. 15. Table 3 shows numerically the average delay values presented in Fig. 16. The solutions based on the 5G stack (i.e., e5G-BDP, DRQL and DynRLC) notably surpass the solutions that are based on other mechanisms (i.e., CoDel and BBR), while the default scenario (i.e., Vanilla) manifestly shows the present bufferbloat problem in cellular networks. Table 4 depicts the 95% CDF value for average delay as shown in Fig. 14, while Table 5 displays the percentage of used RBs in the train and pedestrian scenarios. It can be clearly observed that in both scenarios,

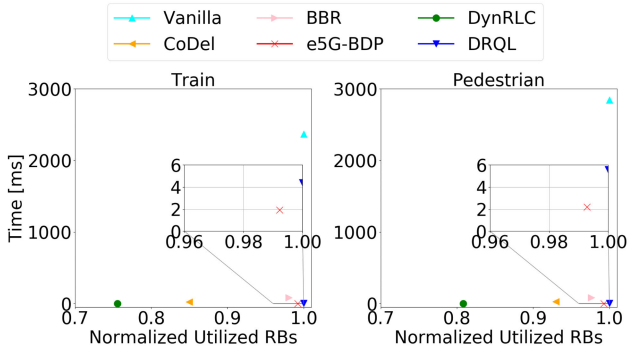

FIGURE 16. Normalized Utilized RBs vs Average Delay for the train and pedestrian datasets.

TABLE 3. Average Delay in ms at the eNodeB for the train and pedestrian scenarios.

	Van.	CoDel	BBR	e5G-BDP	Dyn.	DRQL
Train.	2366.08	24.27	84.06	**1.93**	2.67	4.42
Ped.	2841.73	31.58	85.60	**2.19**	2.92	5.59

TABLE 4. 95% Delay in ms at the eNodeB for the train and pedestrian scenarios.

	Van.	CoDel	BBR	e5G-BDP	Dyn.	DRQL
Train.	5325.93	52.93	186.94	**3.93**	5.56	8.94
Ped.	5261.16	138.94	218.93	**3.90**	5.57	8.77

TABLE 5. Percentage of used RBs for the train and pedestrian scenarios.

	Van.	CoDel	BBR	e5G-BDP	Dyn.	DRQL
Train	**100%**	85.0%	98.1%	99.1%	75.6%	100%
Ped.	**100%**	93.0%	97.6%	99.3%	80.9%	100%

e5G-BDP outperforms in average every competitor in average sojourn time and the 95% delay packet, while squandering less than 1% of the available resources. DynRLC and DRQL show comparable results as DynRLC sacrifices bandwidth, while DRQL suffers larger delays. CoDel and BBR show their limitations, while Vanilla clearly exposes the contemporary bufferbloat problem. From the results provided, we conclude that e5G-BDP is the best solution tested in this paper for addressing the bufferbloat.

B. RLC SUBLAYER PACKET SEGMENTATION/REASSEMBLY

1) STATIC MCS SCENARIO

As analysed in Section IV-B, RLC sublayer's capacity to segment the packets generates an unnecessary, yet avoidable, delay. In this subsection, we evaluated the EQP algorithm against the *Fixed Partition* or FP RB distribution algorithm (i.e., the RBs are scheduled in a fixed manner) in a slicing scenario with the e5G-BDP algorithm as our base bufferbloat avoidance solution. Similar comparative results are expected for DRQL or DynRLC. We created two different slicing scenarios, one where both slices are assigned the 50% of the available resources, and a second one where the resources are shared with a 75%-25% ratio. For both scenarios we generated a low-latency traffic flow along with a bulky traffic flow.

As it can be observed in Fig. 17, the EQP significantly reduces the delay suffered by low-latency flows when compared to the FP resource scheduling. However, the benefit is more evident in the slices with scarce RB share. In a 25% RB slice, approximately 78% of the packets are delivered during the first 1000 μs, in contrast with the approximately 15% when FP is used. The improvement is reduced, albeit is still significant, when the RBs are equally shared, with an approximate 64% vs. 14% of packets being forwarded within the first 1000 μs. Moreover, the results are non-negligible for the slice with 75% of the RBs, where a 88% vs. a 73% is observed. These results confirm the validity of the model

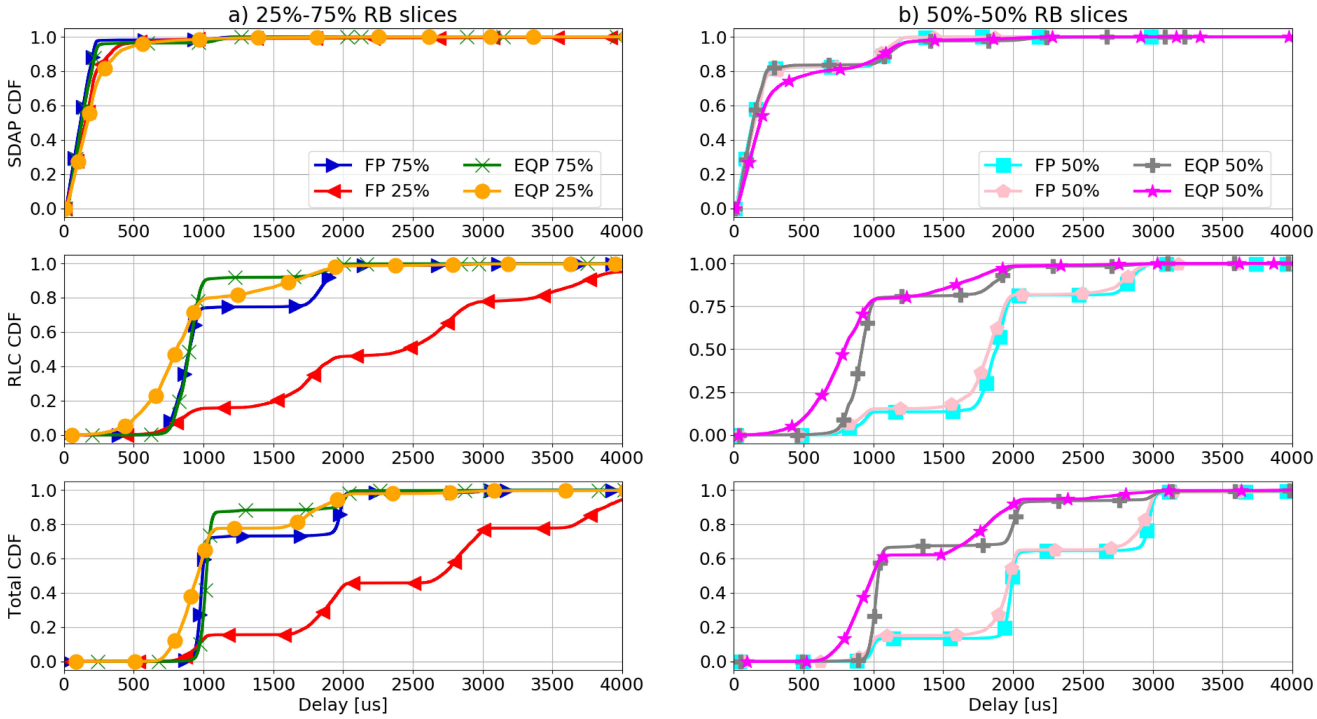

FIGURE 17. CDF of the queuing delay at SDAP, RLC and the sum of both, for FP and EQP in two slicing scenarios: a) 75%-25% and b) 50%-50% resource distribution with 28 MCS.

exposed in Section IV-B. A fixed RB partition does not maximize the number of packets forwarded (i.e., objective function 1), and thus, the delay is larger in comparison with the EQP scheduling.

EQP's objective is to foster the forwarding of full packets to avoid the sojourn time suffered at the UE's RLC when segmentation occurs. This effect can be clearly seen in Fig. 17 where the use of EQP reduces the latency of the VoIP packets for both scenarios. However, EQP squanders bytes when the last packet of the queue is transmitted. For example, 18 RBs can transport up to 1692 bytes with a 28 MCS [20], so if the queue contains 1650 bytes, 42 padding bytes are added into the last RB, and therefore, 42 bytes do not transport information, reducing the throughput. This effect is more pronounced due to the RB distribution based in RBGs [35] (e.g., for a 5 MHz bandwidth cell, the minimum RBG size is 2). Such behavior supposes a challenge for e5G-BDP, as enough packets should be forwarded to the RLC to minimize the buffer starvation, without bloating it. However, the padding effect, as seen in Fig. 18, for 5 MHz bandwidth base stations, does not substantially reduce the throughput, when compared with FP. A 15.80 vs. 15.60, 15.45 vs. 15.35 and 16.20 vs. 15.40 MBits/sec for the 25%-75%, 50%-50% and 75%-25% resource distribution for FP and EQP is reported. OAI's default resource distribution (i.e., FP) discards the last RB (i.e., $\lfloor 0.75 * 25\ RBs \rfloor = 18\ RBs$ and $\lfloor 0.25 * 25\ RBs \rfloor = 6\ RBs$ for the total of 25 RBs), and therefore, the average throughput of Fig. 18 is slightly smaller than the average throughput of e5G-BDP in Fig. 10.

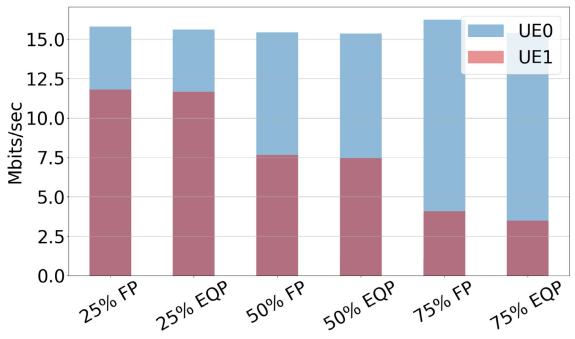

FIGURE 18. Average throughput reported by *iperf3* in two slicing scenarios (i.e., 75%-25% and 50% - 50% resource distribution) for both UEs with 28 MCS for FP and EQP.

The possibility of acquiring more resources than the ones assigned in a strict RB distribution (i.e., FP), emerges here again. As explained in Section IV-B and more concisely in [51], momentarily using more resources than the ones agreed in a SLA, reduces the latency in realistic use case scenarios, while achieving competitive throughputs as seen in the summary of results in Table 6.

2) SCALABILITY EVALUATIONS

As previously mentioned, the cellular network is exposed to services with heterogeneous QoS requirements, that can contain several flows each. To this end, we evaluated 1, 2, 4, and 8 VoIP flows in parallel, to emulate a multi user VoIP call, in a 25%-75% slice resource sharing scenario, where the

TABLE 6. Performance comparison of static MCS scenario: Percentage of packets with <1 *ms* queuing delay, and the throughput of *iperf*3 flow.

	Delay	Throughput
25% EQP	78%	3.50
25% FP	15%	4.10
Ratio (EQP/FP)	**5.20**	**0.85**
50% EQP	64%	7.45
50% FP	14%	7.65
Ratio (EQP/FP)	**4.57**	**0.97**
75% EQP	88%	11.65
75% FP	73%	11,75
Ratio (EQP/FP)	**1.14**	**0.99**

25% slice contains the low-latency flows along a bulky traffic flow, while the 75% slice, only transports one bulky traffic flow. As it can be observed in Fig. 19, EQP considerably surpasses the FP solution, reducing the sojourn time of the low-latency packets. However, a saturation effect is observed when the number of VoIP flows increases. When 8 flows are generated a packet every 2.5 ms is in average created (i.e., a packet is created every 20 ms per flow to emulate a VoIP flow [31]/8 flows $=$ 2.5 *ms*/packet). When the interarrival time of low-latency packets approaches to one TTI (i.e., 1 *ms* in our testbed), the quantum mechanism's advantage is attenuated as EQP's core idea is to momentarily borrow some resources and give them back during the next TTIs. The percentage of packets that are submitted within 2000 μs with EQP are 98%, 98%, 96% and 94% for 1, 2, 4, and 8 VoIP flows while for similar results (i.e., 96%, 95%, 95% and 88% of the packets) the FP needs 4000 μs.

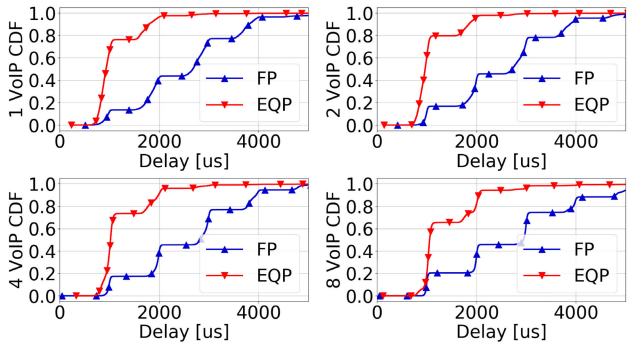

FIGURE 19. Total CDF queuing delay for VoIP packets with 1,2,4 and 8 flows with 28 MCS in a 25% slice for FP and EQP.

3) EFFECT OF QUANTUM LIMIT

As explained in Section IV-B, the quantum limit plays a central role in EQP, as it indicates the amount of RBs that a slice can lend or borrow, thus denoting the maximum RB deviation between the SLA and the slice. To test the effect of the quantum limit within the EQM, we generated two slices with a 25%-75% RB distribution. A bulky flow along with 8 VoIP flows traverse the 25% slice, while only a bulky flow is instantiated at the 75% slice. As observed in Fig. 20, augmenting the quantum limit from 25 to 100 improves the latency of the VoIP packets. However, after a value

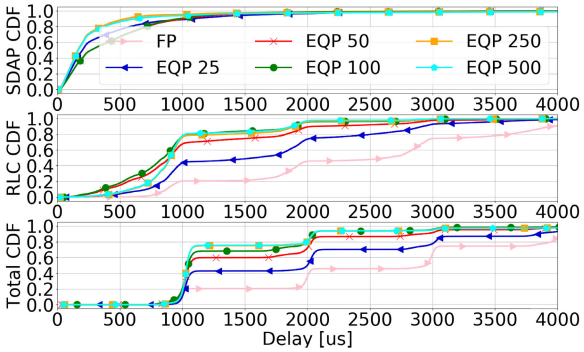

FIGURE 20. Total CDF queuing delay for VoIP packets with 8 flows in a 25% slice with 28 MCS for FP and EQP with 25, 50, 100, 250 and 500 RBs quantum limit.

of 100, augmenting the quantum limit value does not provide any latency reduction. This fact shows that there exists a boundary to the achievable latency reduction for the current packet sequence tested and thus, augmenting the quantum limit value beyond it does not report any benefit. However, even though we tested the quantum value in our most stringent scenario, 5G is an heterogeneous network with myriads of different traffic patterns. Therefore, if the quantum value is to be optimized, the traffic patterns that traverse the slices must be known beforehand. Although it consist in a very interesting problem for contemporary research topics (e.g., machine learning), predicting the future traffic flows lies out of the target of this paper, and therefore is not analyzed.

In Fig. 21 the boxplot for the quantum distribution can be observed. The zero value indicates the equilibrium (i.e., no lent or borrowed RB), while a negative value shows that the slice is indebted, and a positive value manifests that less than the SLA RBs have been used by that slice. The 25% slice tends to borrow resources to forward the VoIP packets, and thus, to maximize the number of forwarded packets, while the 75% slice tends to lend resources as only bulky packets traverse it. Moreover, EQP tends to be a fair RB scheduling (i.e., the median slightly deviates from the equilibrium point) even if large amounts of quantum RBs are available. This is an important property in an environment where the future traffic demands are unknown, as it lets a margin for borrowing RBs in the future if needed, while maintaining the fairness stipulated by the SLA. As expected, the median values of the scenarios with lower quantum values are closer to the equilibrium point in comparison with higher quantum values.

4) DYNAMIC MCS SCENARIO

The previously tested scenarios were based in a static MCS. However, the radio link conditions can abruptly change, and thus, directly impact the throughput. Moreover, the MCS changes do not affect all the UEs equally. For example, one pedestrian carrying a UE and one UE inside a train will be exposed to different MCS profiles. Therefore, in our last scenario, we analyzed two slices with 50% share of the total resources with a bulky flow and a VoIP flow each in

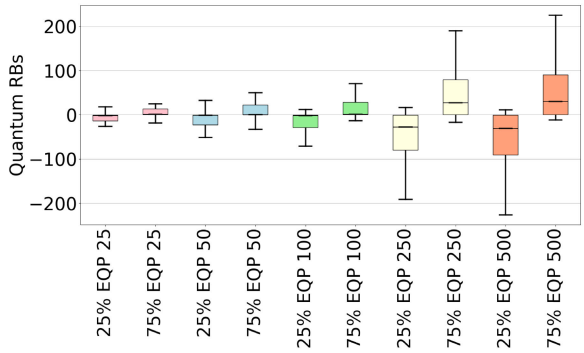

FIGURE 21. Quantum boxplot in a 25%-75% slicing scenario with 8 VoIP flows and a bulky flow in the 25% slice and a bulky flow in the 75% slice.

a pedestrian and train scenario, to realistically validate EQP. In the first slice, a UE (i.e., UE0) with the pedestrian MCS scenario is attached, while for the second slice a UE (i.e., UE1) with the train MCS scenario is utilized. Fig. 22 shows the CDF sojourn time for both slices, where EQP unambiguously surpasses the FP distribution, especially for the pedestrian slice. This occurs since the average MCS for the pedestrian slice is smaller than the average MCS for the train slice (i.e., 11.02 vs. 14.48 on average, cf. Fig. 12) which is strongly connected with the throughput. Therefore and since EQP especially benefits the scenarios with scarce resources, the difference between EQP and FP for the pedestrian slice is more explicit. Within the first 4000 μs, EQP forwards approximately 66% and 76% of the total low-latency packets for the pedestrian and train slices, in contrast with the 44% and 77% with OAI's default approach (i.e., FP). This is an expected result as the throughput is reduced when compared to the static MCS scenario (i.e., a 28 MCS index against the MCS reported in Fig. 12). Here again EQP significantly reduces the latency of the slice with lower throughput (i.e., pedestrian), while achieves similar results for the slice with higher throughput (i.e., train).

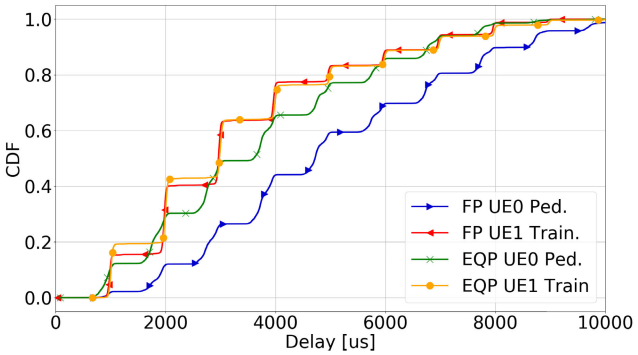

FIGURE 22. Total CDF queuing delay for VoIP packets in two 50%-50% slices for the train and pedestrian datasets for FP and EQP.

VII. CONCLUSION

In this paper we have extensively evaluated RLC buffer delay avoidance solutions in a testbed. The bufferbloat phenomenon in the hierarchical multi-queue 5G QoS architecture has been presented. We verified that 3GPP's Vanilla solution does not address the packet accumulation problem (i.e., the bufferbloat) at RLC's buffer, for which, we presented a solution (i.e., e5G-BDP) and evaluated it against different state-of-the-art proposals. We also exposed RLC's segmentation/reassembly procedure contribution to the latency, presented the EQP algorithm for minimizing such effect and evaluated it against a strict spectrum scheduling algorithm. The most remarkable results from this paper can be summarized as follows:

- Contemporary cellular networks bloat their RLC sublayer buffers, and therefore, ruin 3GPP's efforts to reduce the latency through stack and protocol improvements.
- Introduction of the SDAP sublayer is insufficient if the heterogeneous QoS requirements of different services has to be fulfilled. 3GPP should consider the insertion of a queue-based architecture in the SDAP sublayer with scheduling capabilities to avoid bloating its stack and achieve its low-latency envisioned goals.
- Classic bufferbloat solutions (e.g., CoDel and BBR) do not suffice to meet the most stringent latency requirements in 5G. New solutions that directly rely on the cellular network stack and gather information from it must be considered as they outperform the classic solutions by an order of magnitude (e.g., e5G-BDP).
- In a packet based network such as the cellular network, packet sizes must be considered by the resource scheduling algorithm, if latency is a concern. Solutions ignoring such fact only generate a myriad of segmented packets at the RLC sublayer increasing the total latency and impeding a rapid packet delivery. We presented, evaluated and validated an algorithm (i.e., EQP) that reduces such effect, while deviating from the SLA within a limit.

As future work, we plan to study the wired network delays (e.g., Time Sensitive Networks) in conjunction with the 5G stack delays.

REFERENCES

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, Oct. 1948, pp. 379–423.
[2] S.-Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.
[3] *NR, Multiplexing and Channel Coding*, 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 3GPP, document 38.212, version 16.1.0., Apr. 2020.
[4] *Study on Physical Layer Enhancements for NR Ultra-Reliable and Low Latency Case (URLLC)*, 3rd Generation Partnership Project (3GPP), Technical Specification (TS) document 38.824, version 15.5.1., Mar. 2019.
[5] G. Berardinelli, N. H. Mahmood, R. Abreu, T. Jacobsen, K. Pedersen, I. Z. Kovacs, and P. Mogensen, "Reliability analysis of uplink grant-free transmission over shared resources," *IEEE Access*, vol. 6, pp. 23602–23611, 2018.
[6] *Evolved Universal Terrestrial Radio Access (E-UTRA) and NR; Service Data Adaptation Protocol (SDAP) Specification*, 3rd Generation Partnership Project (3GPP), Technical Report (TR) 3GPP, document 37.324, version 15.1.0., Sep. 2018.
[7] *System Architecture for the 5G System*, 3rd Generation Partnership Project (3GPP), Technical Report (TR) document 23.501, version 15.4.0., Dec. 2018.

[8] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008, doi: 10.1145/1400097.1400105.

[9] *The Bufferbloat Project*. Accessed: Dec. 1, 2020. [Online]. Available: https://www.bufferbloat.net/projects/

[10] T. Høiland-Jørgensen, M. Kazior, D. Täht, P. Hurtig, and A. Brunstrom, "Ending the anomaly: Achieving low latency and airtime fairness in WiFi," in *Proc. Annu. Tech. Conf. (USENIX-ATC)*. Santa Clara, CA, USA: USENIX Association, 2017, pp. 139–151.

[11] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, *Controlled Delay Active Queue Management*, Internet Requests for Comments, RFC Editor, document RFC 8289, Jan. 2018.

[12] T. Hoeiland-Joergensen, P. McKenney, D. Taht, J. Gettys, and E. Dumazet, *The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm*, Internet Requests for Comments, RFC Editor, RFC document 8290, Jan. 2018.

[13] *NR, Radio Link Control (RLC) Specification*, 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 3GPP, document 38.322, version 15.4.0., Jan. 2019.

[14] *LTE, Radio Link Control (RLC) Protocol Specification*, 3rd Generation Partnership Project (3GPP), Technical Specification (TS) document 36.322, version 16.0.0., Jul. 2020.

[15] P. Caballero, A. Banchs, G. de Veciana, and X. Costa-Perez, "Multi-tenant radio access network slicing: Statistical multiplexing of spatial loads," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3044–3058, Oct. 2017.

[16] F. Fossati, S. Moretti, P. Perny, and S. Secci, "Multi-resource allocation for network slicing," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1311–1324, Jun. 2020.

[17] L. Kleinrock, "Internet congestion control using the power metric: Keep the pipe just full, but no fuller," *Ad Hoc Netw.*, vol. 80, pp. 142–157, Nov. 2018.

[18] *Interface Between the Control Plane and the User Plane Nodes*, 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 3GPP, document 29.244, version 16.0.0., Jun. 2019.

[19] *NR, Radio Resource Control (RRC) Protocol Specification*, 3rd Generation Partnership Project (3GPP), Technical Specification (TS) document 38.331, version 15.5.1., Apr. 2019.

[20] *Open Air Interface Feature Set*. Accessed: Dec. 1, 2020. [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/blob/develop/doc/FEATURE_SET.md

[21] M. Irazabal, E. Lopez-Aguilera, and I. Demirkol, "Active queue management as quality of service enabler for 5G networks," in *Proc. Eur. Conf. Netw. Commun. (EuCNC)*, Jun. 2019, pp. 1–5.

[22] R. Kumar, A. Francini, S. Panwar, and S. Sharma, "Dynamic control of RLC buffer size for latency minimization in mobile RAN," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6.

[23] R. Kumar, A. Francini, S. Panwar, and S. Sharma, "Design of an enhanced bearer buffer for latency minimization in the mobile RAN," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[24] M. Irazabal, E. Lopez-Aguilera, I. Demirkol, and N. Nikaein, "Dynamic buffer sizing and pacing as enablers of 5G low-latency services," *IEEE Trans. Mobile Comput.*, early access, Aug. 17, 2020, doi: 10.1109/TMC.2020.3017011.

[25] F. Baker and G. Fairhurst, *IETF Recommendations Regarding Active Queue Management*, document IETF 7567, Internet Requests for Comments, RFC Editor, RFC, Jul. 2015.

[26] S. Jung, J. Kim, and J.-H. Kim, "Intelligent active queue management for stabilized QoS guarantees in 5G mobile networks," *IEEE Syst. J.*, early access, Aug. 17, 2020, doi: 10.1109/JSYST.2020.3014231.

[27] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *Commun. ACM*, vol. 60, no. 2, pp. 58–66, Jan. 2017.

[28] P. Goyal, M. Alizadeh, and H. Balakrishnan, "Rethinking congestion control for cellular networks," in *Proc. 16th ACM Workshop Hot Topics Netw.* New York, NY, USA: Association for Computing Machinery, Nov. 2017, pp. 29–35, doi: 10.1145/3152434.3152437.

[29] P. E. McKenney, "Stochastic fairness queueing," in *Proc. 9th Annu. Joint Conf. IEEE Comput. Commun.*, vol. 2, Jun. 1990, pp. 733–740.

[30] B. Briscoe, K. D. Schepper, M. B. Braun, and G. White. (Mar. 2020). *Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service* Internet Engineering Task Force, Internet-Draft draft-ietf-tsvwg-l4s-arch-06. work in Progress. [Online]. Available: https://tools.ietf.org/html/draft-ietf-tsvwg-l4s-arch-06

[31] *VoIP Bandwidth Consume*. Accessed: Dec. 1, 2020. [Online]. Available: https://www.cisco.com/c/en/us/support/docs/voice/voice-quality/7934-bwidth-consume.html

[32] *Google Stadia*. Accessed: Dec. 1, 2020. [Online]. Available: https://stadia.google.com/

[33] *Open Air Interface*. Accessed: Dec. 1, 2020. [Online]. Available: https://www.openairinterface.org/

[34] *srsLTE*. Accessed: Dec. 1, 2020. [Online]. Available: https://github.com/srsLTE/srsLTE

[35] *NR; Physical Layer Procedures for Data*, 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 3GPP, document 38.214, version 15.7.0., Sep. 2019.

[36] C.-Y. Chang and N. Nikaein, "RAN runtime slicing system for flexible and dynamic service execution environment," *IEEE Access*, vol. 6, pp. 34018–34042, 2018.

[37] H. Halabian, "Distributed resource allocation optimization in 5G virtualized networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 627–642, Mar. 2019.

[38] R. Schmidt, C. Chang, and N. Nikaein, "FlexVRAN: A flexible controller for virtualized RAN over heterogeneous deployments," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[39] C. Hornig, *A Standard for the Transmission of IP Datagrams Over Ethernet Networks*, document RFC 894, Internet Requests for Comments, RFC Editor, RFC, Apr. 1984.

[40] *IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks–Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11-2016 (Revision of IEEE Standard 802.11-2012), 2016, pp. 1–3534.

[41] C. A. Grazia, N. Patriciello, T. Hoiland-Jorgensen, M. Klapez, M. Casoni, and J. Mangues-Bafalluy, "Adapting TCP small queues for IEEE 802.11 networks," in *Proc. IEEE 29th Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2018, pp. 1–6.

[42] NR, *Physical Channels and Modulation*, 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 3GPP, document 38.211, version 15.6.0., Jun. 2019.

[43] *Study on New Radio Access Technology: Radio Access Architecture and Interfaces*, 3rd Generation Partnership Project (3GPP), Technical report (TR) document 38.801, version 14.0.0., Apr. 2017.

[44] M. Mondal, B. Roy, C. K. Roy, and K. A. Schneider, "Investigating the relationship between evolutionary coupling and software bug-proneness," in *Proc. 29th Annu. Int. Conf. Comput. Sci. Softw. Eng.* Armonk, NY, USA: IBM Corp., 2019, pp. 173–182.

[45] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "OpenAirInterface: A flexible platform for 5G research," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 33–38, Oct. 2014.

[46] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 5th ed. Cham, Switzerland: Springer, 2016.

[47] D. S. Johnson and K. A. Niemi, "On knapsacks, partitions, and a new dynamic programming technique for trees," *Math. Oper. Res.*, vol. 8, no. 1, pp. 1–14, Feb. 1983.

[48] *Dbs3900 Huawei Distributed Base Stations*. Accessed: Dec. 1, 2020. [Online]. Available: https://e.huawei.com/en/products/wireless/elte-trunking/network-element/dbs3900

[49] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, New York, NY, USA, 1995, pp. 231–242.

[50] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *IEEE/ACM Trans. Netw.*, vol. 1, no. 3, pp. 344–357, Jun. 1993.

[51] M. H. Goldwasser, "A survey of buffer management policies for packet switches," *ACM SIGACT News*, vol. 41, no. 1, pp. 100–128, Mar. 2010, doi: 10.1145/1753171.1753195.

[52] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[53] W. A. Aiello, Y. Mansour, S. Rajagopolan, and A. Rosén, "Competitive queue policies for differentiated services," *J. Algorithms*, vol. 55, no. 2, pp. 113–141, May 2005.

[54] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: A 4G LTE dataset with channel and context metrics," in *Proc. 9th ACM Multimedia Syst. Conf.*, New York, NY, USA, Jun. 2018, pp. 460–465.

[55] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, "FlexRAN: A flexible and programmable platform for software-defined radio access networks," in *Proc. 12th Int. Conf. Emerg. Netw. Exp. Technol.* New York, NY, USA: Association for Computing Machinery, Dec. 2016, pp. 427–441, doi: 10.1145/2999572.2999599.
[56] *IRTT Isochronous Round-Trip Tester*. [Online]. Available: https://github.com/heistp/irtt
[57] M. Belshe, R. Peon, and M. Thomson, *Hypertext Transfer Protocol Version 2 (HTTP/2)*, Internet Requests for Comments, RFC Editor, RFC document 7540, 2015.

MIKEL IRAZABAL received the M.Sc. degree in telecommunications engineering from the Universitat Politècnica de Catalunya (UPC), Barcelona, in 2011, where he is currently pursuing the Ph.D. degree with the Department of Network Engineering. He participated as an Early Stage Researcher (ESR) in the European funded Project Application-Aware User-Centric Programmable Architectures for 5G multi-tenant networks (5G-AuRA) and an Innovative Training Network (ITN) of the Marie Skłodowska-Curie Actions (MSCA). His research interests include the low-latency wireless communications and programmable wireless communication systems.

ELENA LOPEZ-AGUILERA received the M.Sc. and Ph.D. degrees in telecommunications engineering from the Universitat Politècnica de Catalunya (UPC), in 2001 and 2008, respectively. She is currently an Associate Professor and a member with the Wireless Networks Group, Networks Engineering Department, UPC. She has published articles in journals and conferences in the area of wireless communications and has been involved in projects with public and private funding. Her research interests include the study of the IoT enabling technologies and 5G networks. Her experience comprises QoS, radio resource management, location mechanisms, and wake-up radio systems.

ILKER DEMIRKOL received the B.Sc., M.Sc., and Ph.D. degrees in computer engineering from Boğaziçi University, İstanbul, Turkey. He is currently an Associate Professor with the Department of Mining, Industrial and ICT Engineering, Universitat Politècnica de Catalunya, where he works on wireless networks and wake-up radio systems. His research interests include communication protocol development for the aforementioned networks, along with performance evaluation and optimization of such systems. He was a recipient of the 2010 Best Mentor Award from the Electrical and Computer Engineering Department, University of Rochester, Rochester, NY, USA.

ROBERT SCHMIDT (Graduate Student Member, IEEE) received the Diploma (Hons.) in information systems engineering from the Dresden University of Technology, Germany, and the Diploma in engineering from the Ecole Centrale Paris/CentraleSupélec, France, in 2017. He is currently pursuing the Ph.D. degree in communications with the Communication Systems Department, EURECOM, France. He is involved in collaborative research projects in the context of the EU H2020 framework programme and an active contributor to the OpenAirInterface and Mosaic5G projects. His research interests include 4G and 5G wireless cellular networks, heterogeneous software-defined (radio access) networks, network slicing, and MAC layer scheduling.

NAVID NIKAEIN received the Ph.D. degree in communication systems from the Swiss Federal Institute of Technology (EPFL), in 2003. He is currently a Professor with the Communication System Department, EURECOM. Broadly, his research interests include experimental 4G-5G system research related to radio access, edge, and core networks with a blend of communication, computing, and data analysis with a particular focus on industry-driven use-cases. He is also a Board Member of OpenAirInterface Software Alliance as well as the Founder of the Mosaic-5G.io. initiative whose goal is to provide software-based 4G/5G service delivery platforms.