

# Ph.D. Thesis in Network Engineering

## Title:

# Enhanced Quality of Service mechanisms for 5G networks

**Ph.D. Candidate:** Mikel Irazabal

**Ph.D. Advisors:** Dr. Elena Lopez-Aguilera and Dr. Ilker Demirkol



**UNIVERSITAT POLITÈCNICA DE CATALUNYA**  
**BARCELONATECH**

**Departament d'Enginyeria Telemàtica**

May, 2021



# Abstract

The heterogeneous services with stringent requirements that are envisioned to co-exist under the fifth generation of cellular networks (5G), inexorably challenge the current Long-Term Evolution (LTE) cellular network's features. Consequently an important amount of effort has been invested to reduce current latency while increasing the throughput and the reliability. As an example, 5G's uplink granted free transmission procedure permits reserving Resource Blocks (RBs) in advance for a set of User Equipments (UEs), and thus, reduce the latency by eliminating the UEs resource request procedure. Likewise, the cellular network stack has experience major changes in different layers. Examples of it are the new subcarrier spacing that allow reducing the slot duration, and thus, the transmission time or the new Radio Link Controller (RLC) Packet Data Unit (PDU) that enables a faster packet forming at the expense of reducing the compression ratio. These solutions address latency causes that lie in 5G's protocol stack, and thus, endogenous, and will unquestionably enhance 5G's capability to meet the rigorous services' latency requirements. In addition, 5G has introduced a new sublayer (i.e., Service Data Adaptation Protocol (SDAP)), and has defined a new Quality of Service (QoS) Flow Indicator (QFI) as its finest quality granularity indicator. However, delays generated by the pace at which data packets are forwarded or its' sizes, that can be classified as exogenous causes since they do not depend on 5G's specifications, can significantly impact the latency in contemporary cellular networks, and ruin the stringent timing guarantees required by delay-sensitive services if they are not carefully considered.

One of the problems associated with the pace at which the data packets are forwarded is the bufferbloat, and will specifically occur in 5G's Radio Access Network RAN since contemporary wired links are orders of magnitude faster than wireless links, and are provided with large buffers to always fulfill the fluctuating radio link capacity, and thus, avoid squandering wireless resources.

Being the RAN the bottleneck and having large buffers together with the fact that most of contemporary data is transported through the loss-based congestion control Transmission Control Protocol (TCP) Cubic, suffice to bloat the buffers and considerably increase the latency. In this thesis we provide quantitative results of the bufferbloat problem in contemporary cellular networks. We first thoroughly explain 5G's QoS hierarchical multi-queuing and show how the

bufferbloat effect significantly increases the latency. Following, we propose the (enhanced) 5G Bandwidth Delay Product ((e)5G-BDP), the Dynamic RLC Queue Limit (DRQL) and the UPF-SDAP Pacer (USP) solutions, enhancing the current 5G QoS multi-queuing architecture, and approaching through two different paradigms to the bufferbloat problem. We evaluate our proposed solutions in an emulator, as well as in a testbed, against state-of-the-art solutions and conclude that a new QoS hierarchical multi-queuing architecture is needed in 5G to fulfill the latency requirements for which it is envisioned.

Moreover, since information in the wired link is transported in packets, while in the RAN is transmitted through RBs, packets that do not fit in the assigned RBs, are segmented and transmitted during different Transmission Time Intervals (TTIs). Such mechanism prevents wasting the scarce wireless transmission opportunities. However, the segmented information at the receiver cannot be forwarded until all the remaining information from the packet is reassembled, which in the best case occurs during the next transmission opportunity. We exhaustively study the problem and propose a RB scheduling algorithm named Elastic Quantum Partition (EQP) to address this challenge and compare it quantitatively against a Fixed Partition (FP) RB distribution in different scenarios in a testbed with dynamic Modulation and Coding Scheme (MCS), off-the-shelf equipment and slices. The outcome shows a latency reduction when scheduling the RBs elastically rather than using a fixed scheduler.

In summary, in this thesis we shed some light in the bufferbloat phenomenon and the segmentation/reassembly procedure in current cellular networks. We propose and evaluate novel solutions that mitigate both effects, ultimately reducing the latency in the current cellular network.

# Acknowledgments

This thesis is not just the culmination of my research work during the last years, but also the outcome of a harsh struggle against the prevailing despotism and injustice that mostly reigns in the academic/research kingdom. In this regard I want to specially thank Nikos and Matteo for being incredible people. They never let me down in the most difficult moments, and I had the pleasure to share special moments with them that will endure in my memories forever. They were present when for my first time in life I met a lawyer, and when the EU, AGAUR, CTTC and UPC turned a blind eye in structural problems. Thanks for being firm and not abandoning the ship of justice and truth. However and luckily, humor is tragedy plus time<sup>1</sup> and therefore, we will most likely remember that moments with joy and irony rather than the anger it legitimately deserves. *Es setzen sich nur so viel Wahrheiten durch, als wir durchsetzen; der Sieg der Vernunft kann nur der Sieg der Vernünftigen sein*<sup>2</sup>.

I would also like to thank my advisors Dr. Elena Lopez-Aguilera and Dr. Ilker Demirkol for their support and guidance during this thesis. Without their help this thesis would not had been born.

All of this work would of course have not been possible without my family, and especially, my parents and my sister's support. I feel grateful for their life support and the moments that we shared together during this thesis, even though I would have liked to share more time with you.

I would also like to thank the Marie Skłodowska-Curie Actions and specifically the 5G-AuRA project (No. 675802) for their economical support. 5G-AuRA also gave me the possibility to meet young researchers of whom I would like to mention: Matteo, Nikos, Akshay, Ronny, Girma, Lanfranco, Kafi, Tareq, Jian, Meysam and Asif. I also had the chance to fulfill a research secondment at EURECOM under the supervision of Dr. Navid Nikaein to which I am also very grateful as he guided me while discovering OpenAirInterface's guts.

Finding a flat in Barcelona is at best, a nightmare. Therefore, thank you very much Christoph for letting me stay with you at La Floresta, sharing many moments and teaching me the art of

---

<sup>1</sup>M. Twain.

<sup>2</sup>B. Brecht.

tomato growing.

Of course this section wouldn't be finished without a special thanks to Vincenza. She is the person that best knows the miseries and glories that I went through. Her support during these years has been crucial for my personal and academic development. In fact, you have made this journey much more livable. Thank you.

Lastly, I want to thank all my friends and people (which of course, I am going to omit the list as it would be *incomplet* and *inkorrek*t) I met during this journey that started at my always loved city Vitoria-Gasteiz. Whether I met them at Vitoria-Gasteiz, Barcelona, Berlin or somewhere else I want to thank you all for your support.

# List of Contributions

The following publications were produced while elaborating this thesis.

## Journals:

[J1] M. Irazabal, E. Lopez-Aguilera, I. Demirkol and N. Nikaein, "Dynamic Buffer Sizing and Pacing as Enablers of 5G Low-Latency Services," (Early access) in *IEEE Transactions on Mobile Computing*. (Area: Telecommunications; Quartile Q1; IF: 5.112).

[J2] M. Irazabal, E. Lopez-Aguilera, I. Demirkol, R. Schmidt and N. Nikaein, "Preventing RLC Buffer Sojourn Delays in 5G," in *IEEE Access*, vol. 9, pp. 39466-39488, March 2021. (Area: Telecommunications; Quartile Q1; IF: 3.745).

## Conferences:

[C1] M. Irazabal, E. Lopez-Aguilera, and I. Demirkol, "Active Queue Management as Quality of Service Enabler for 5G Networks," in *2019 European Conference on Networks and Communication (EuCNC)*, April 2019, pp. 1-5.





---

# Contents

---

<b>List of Figures</b>	<b>ix</b>
<b>List of Tables</b>	<b>xiii</b>
<b>List of Abbreviations</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Objectives and Contributions . . . . .	4
1.2 Thesis Outline . . . . .	6
<b>2 Background: The 5G stack QoS model and the Bufferbloat</b>	<b>9</b>
2.1 5G Stack and its QoS Model . . . . .	9
2.2 Bufferbloat Solutions and Delay-Sensitive Enablers . . . . .	17
2.3 Other 5G Low-Latency Enablers . . . . .	21
2.4 Summary . . . . .	22
<b>3 Addressing the bufferbloat phenomenon in 5G</b>	<b>25</b>

3.1	State of the Art in 5G Active Queue Management and Bufferbloat Avoidance Solutions . . . . .	26
3.2	Linux Traffic Control Approaches in the 5G Stack: DRQL . . . . .	29
3.3	A Queuing Theory Approach to the 5G Stack: 5G-BDP . . . . .	33
3.4	Enhanced 5G-BDP: e5G-BDP . . . . .	37
3.5	Pacing the Traffic between the UPF and the SDAP: UPS . . . . .	44
3.6	Evaluation Frameworks and Experimental Results . . . . .	47
3.6.1	5G QoS Hierarchical Queuing Emulator . . . . .	47
3.6.2	5G QoS's Emulator Experimental Results . . . . .	53
3.6.3	Cellular Network Testbed: OAI . . . . .	64
3.6.4	Cellular Network Testbed's Experimental Results . . . . .	68
3.7	Summary . . . . .	77
<b>4</b>	<b>Addressing the latency generated by the RLC sublayer segmentation/reassembly procedure . . . . .</b>	<b>79</b>
4.1	RLC Sublayer . . . . .	80
4.2	RLC Stack Changes in 5G . . . . .	82
4.3	Segmentation/Reassembly Procedure in LTE and 5G . . . . .	86
4.4	Avoiding the Segmentation/Reassembly Procedure: EQP . . . . .	87
4.5	Evaluation Framework and Experimental Results . . . . .	97
4.5.1	Evaluation Framework . . . . .	97
4.5.2	Experimental Results . . . . .	98
4.6	Summary . . . . .	104
<b>5</b>	<b>Conclusions and Future Work . . . . .</b>	<b>107</b>
5.1	Conclusions . . . . .	108
5.2	Future Work . . . . .	110
	<b>Bibliography . . . . .</b>	<b>113</b>

---

## List of Figures

---

2.1	3GPP SBA. From 3GPP Technical Specification (TS) 23.501 [1]. . . . .	11
2.2	5G's RAN stack. From 3GPP TS 38.300 [2]. . . . .	12
2.3	5G's QoS scenario. From 3GPP TS 23.501 [1]. . . . .	13
2.4	Simplified 5G QoS downlink block diagram with the entities, buffers, QoS flow and DRB abstractions. . . . .	13
2.5	SDAP functional view. From 3GPP 37.324 [3]. . . . .	14
2.6	SDAP structural view. From 3GPP 37.324 [3]. . . . .	15
2.7	5G's Layer 2 SDAP, PDCP, RLC and MAC sublayers in downlink procedure. From 3GPP TS 38.300 [2]. . . . .	16
3.1	Simplified Linux network stack. . . . .	30
3.2	Simplified data path model for SDAP and RLC sublayers queues with multiple QFIs that converge to one DRB. . . . .	34
3.3	A D/D/K deterministic queuing system. . . . .	34
3.4	Possible e5G-BDP pacing mechanism outcome. . . . .	43
3.5	Evaluation framework with first and second scenarios. . . . .	49
3.6	Average queue size and average delay for pedestrian dataset. . . . .	54

3.7	Radio link channel capacity and RLC queue occupancy. . . . .	55
3.8	First scenario and pedestrian dataset: utilization rate and delay. . . . .	56
3.9	First scenario and train dataset: utilization rate and delay. . . . .	57
3.10	Low-latency traffic boxplot for pedestrian dataset and interval times of 20, 40, 60 and 70 ms. . . . .	58
3.11	Low-latency traffic boxplot for train dataset and interval times of 20, 40, 60 and 70 ms. . . . .	59
3.12	Delay vs. normalized bandwidth with two low-latency flows and a bulky TCP flow with pedestrian dataset. . . . .	60
3.13	Delay vs. normalized bandwidth with two low-latency flows and a bulky TCP flow with train dataset. . . . .	60
3.14	Second scenario and pedestrian dataset: utilization rate and delay. . . . .	61
3.15	Second scenario and train dataset: utilization rate and delay. . . . .	62
3.16	Scalability: Delay vs. normalized bandwidth with two low-latency flows and a bulky TCP flow with pedestrian dataset . . . . .	63
3.17	Scalability: Delay vs. normalized Bandwidth with two low-latency flows and a bulky TCP flow with train dataset . . . . .	63
3.18	Evaluation testbed. . . . .	65
3.19	Delay suffered by a VoIP flow when sharing the RLC buffer with a bulky flow in Vanilla OAI. . . . .	68
3.20	RLC buffer occupancy for different methods for MCS index of 28 and 25 RBs. . . . .	69
3.21	Cumulative Distribution Function (CDF) of VoIP flow's RTT. . . . .	70
3.22	CDF of the queuing delay at RLC, SDAP and the sum of both for a MCS index of 28 and 25 RBs. . . . .	71
3.23	Bandwidth reported by <i>iperf3</i> with MCS=28 and 25 RBs. . . . .	72
3.24	Delay faced by VoIP packets with 1, 2, 4 and 8 flows. . . . .	73
3.25	Amount of bytes requested by the MAC sublayer and MCS employed. . . . .	73
3.26	RLC buffer occupancy for different methods for train dataset and 25 RBs. . . . .	74
3.27	CDF for train scenario and 25 RBs. . . . .	75
3.28	Bandwidth for the train scenario. . . . .	75
3.29	Normalized bandwidth vs average delay for train and pedestrian scenario. . . . .	75
4.1	RLC TM from 3GPP 38.322 [4] . . . . .	81

---

4.2	RLC UM from 3GPP 38.322 [4]. . . . .	81
4.3	RLC AM from 3GPP 38.322 [4]. . . . .	82
4.4	RLC UM PDU with 10 bit SN (Odd number of LIs, i.e. $K = 1, 3, 5, \dots$ ) from 3GPP 36.322 [5]. . . . .	83
4.5	RLC UM PDU with 12 bit SN and with SO from 3GPP 38.322 [4]. . . . .	83
4.6	Possible C++ code for accessing RLC's header information in 5G. 8 bytes are needed to indicate the absolute position of 4 SDUs. . . . .	84
4.7	Intel assembler code produced by GCC 10.2 with the optimization flag -O3 for 5G's RLC extended header. . . . .	84
4.8	Possible C++ code for accessing RLC's header information in LTE. 6 bytes are needed to indicate the relative position of 4 SDUs. . . . .	85
4.9	Intel assembler code produced by GCC 10.2 with the optimization flag -O3 for LTE's RLC extended header. . . . .	86
4.10	Illustration of FP and EQP RB distribution algorithms. . . . .	88
4.11	RLC DRB buffers with different sized packets. The red, green and blue colors represent 1, 2 and 3 RB sized packets, respectively. The arrow indicates the precedence constraint between packets. . . . .	89
4.12	CDF of the queuing delay for VoIP packets at SDAP, RLC and the sum of both for FP and EQP in two slicing scenarios (i.e., a) 75%-25% and b) 50%-50% resource distribution) with 28 MCS. . . . .	99
4.13	Average throughput reported by <i>iperf3</i> in two slicing scenarios (i.e., 75%-25% and 50% - 50% resource distribution) for both UEs with 28 MCS for FP and EQP. . . . .	100
4.14	8 VoIP flows scalability for 25% RBs using e5G-BDP. . . . .	101
4.15	CDF of the queuing delay at SDAP, RLC, and the sum of both 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. . . . .	102
4.16	Quantum boxplot in a 25%-75% slicing scenario with 8 VoIP flows, a bulky flow in the 25% slice, and a bulky flow in the 75% slice. . . . .	103
4.17	Total CDF queuing delay for VoIP packets, in two 50%-50% slices for the train and pedestrian datasets, for FP and EQP. . . . .	104



---

## List of Tables

---

3.1	Algorithms tested to validate the bufferbloat avoidance solutions at the emulator.	51
3.2	Average Delay in ms at eNodeB for the train and pedestrian scenarios. . . . .	76
3.3	95% Delay in ms at eNodeB for the train and pedestrian scenarios. . . . .	76
3.4	Percentage of used RBs for the train and pedestrian scenarios. . . . .	76





---

## List of Abbreviations

---

**3GPP** Third Generation of Partnership Project

**5G** Fifth Generation of Cellular Networks

**5G-BDP** 5G Bandwidth Delay Product

**ACK** Acknowledgement

**AF** Application Function

**AM** Acknowledge Mode

**AMF** Access and Mobility Management Function

**AQM** Active Queue Management

**AUSF** Authentication Server Function

**BBandwidth** Bottleneck Bandwidth

**BBR** Bottleneck Bandwidth and Round-trip time

**BCCH** Broadcast Control Channel

**BDP** Bandwidth Delay Product

**BQL** Byte Queue Limit

**BtIBw** Bottleneck Bandwidth

**C2TCP** Cellular Controlled delay TCP

**CCCH** Common Control Channel

**CDF** Cumulative Distribution Function

**CN** Core Network

**CoDel** Controlled Delay

**COTS** Commercial off-the-shelf

**CQI** Channel Quality Indicator

**CU** Central Unit

**DCCH** Dedicated Control Channel

**DiffServ** Differentiated Services

**DN** Data Network

**DQL** Dynamic Queue Limit

**DRB** Data Radio Bearer

**DRQL** Dynamic RLC Queue Limit

**DRR** Deficit Round Robin

**DTCH** Dedicated Traffic Channel

**DU** Distributed Unit

**DynRLC** Dynamic RLC

**E** Extension Bit

**e5G-BDP** enhanced 5G Bandwidth Delay Product

**ECN** Explicit Congestion Notification

**eMBB** enhanced Mobile Broadband

**eNodeB** evolved Node B

**EQP** Elastic Quantum Partition

**EWMA** Exponentially Weighted Moving Average

**FAR** Forwarding Action Rule

**FI** Framing Info

**FIFO** First In First Out

**FP** Fixed Partition

**FQ-CoDel** Flow Queue CoDel

**GaaS** Gaming as a Service

**GBR** Guaranteed Bit Rate

**GPRS** General Packet Radio Service

**GTP** GPRS Tunneling Protocol

**HARQ** Hybrid Automatic Repeat Request

**HSS** Home Subscriber Server

**HTTP/2** Hypertext Transfer Protocol 2

**IEEE** Institute of Electrical and Electronics Engineers

**IoT** Internet of Things

**IP** Internet Protocol

**IPv4** Internet Protocol version 4

**irtt** Isochronous Round-Trip Tester

**LDCP** Low Density Parity Check

**LI** Length Indicator

**LTE** Long Term Evolution

**MAC** Media Access Control

**MAR** Multi-Access Rule

**MCS** Modulation and Coding Scheme

**MEC** Mobile Edge Computing

**mMTC** massive Machine Type Communication

**MTU** Maximum Transmission Unit

**NACK** Negative Acknowledgement

**NEF** Network Exposure Function

**NIC** Network Interface Controller

**NLDelay** No-Load Delay

**Non-GBR** Non-Guaranteed Bit Rate

**NRF** Network Repository Function

**NSSF** Network Slice Selection Function

**OAI** Open Air Interface

**PC** Personal Computer

**PCCH** Paging Control Channel

**PCF** Policy Control Function

**PCKP** Precedence Constrained Knapsack Problem

**PCP** Packet Control Protocol

**PDB** Packet Delay Budget

**PDCP** Packet Data Convergence Protocol

**PDR** Packet Detection Rule

**PDU** Packet Data Unit

**PER** Packet Error Rate

**PHY** Physical Layer

**PIE** Proportional Integral controller Enhanced

**QAM** Quadrature Amplitude Modulation

**QER** QoS Enforcement Rule

**QFI** QoS Flow Indicator

**QoS** Quality of Service

**QPSK** Quadrature Phase Shift Keying

**RAN** Radio Access Network

**RB** Resource Blocks

**RBG** Resource Block Groups

**RED** Random Early Detection

**RFC** Request For Comments

**RLC** Radio Link Control

**RNTI** Radio Network Temporal Identifier

**RRC** Radio Resource Control

**RTprop** Round Trip propagation

**RTT** Round Trip Time

**SBA** Service Based Architecture

**SCTP** Stream Control Transmission Protocol

**SDAP** Service Data Adaptation Protocol

**SDN** Software Defined Network

**SDU** Service Data Unit

**SFQ** Stochastic Fair Queuing

**SLA** Service Level Agreement

**SMF** Session Management Function

**SN** Sequence Number

**SO** Segmentation Offset

**TCP** Transmission Control Protocol

**TS** Technical Specification

**TSN** Time Sensitive Networking

**TSO** TCP Segmentation Offload

**TSQ** TCP Small Queue

**TTI** Transmission Time Interval

**UDM** Unified Data Management

**UDP** User Datagram Protocol

**UE** User Equipment

**UM** Unacknowledge Mode

**UPF** User Plane Function

**URLLC** Ultra Reliable Low-Latency Communication

**URR** Usage Reporting Rules

**USB** Universal Serial Bus

**USP** UPF-SDAP Pacer

**USRP** Universal Software Radio Peripheral

**VLAN** Virtual Local Area Network

**VoIP** Voice over IP

**Wi-Fi** Wireless Fidelity





# CHAPTER 1

---

## Introduction

---

Contemporary zeitgeist cannot be understood without the ubiquitous mobile connectivity and services that modern cellular networks provide. Additionally, a myriad of new business opportunities that rely on services with increased throughput and reliability, and reduced latency are foreseen. To fulfill such requirements, the Third Generation of Partnership Project (3GPP), which encompasses seven telecommunications standard development organizations is performing a substantial effort standardizing the new fifth generation of cellular networks or 5G. 5G is envisioned as the pillar technology that will enable the management of heterogeneous services with different reliability, bandwidth and latency constraints under the same cellular network stack. Since all the services do not require the same treatment, 3GPP has proposed a taxonomy of the services in three groups: Enhanced Mobile Broadband (eMBB), Massive Machine Type Communications (mMTC) and Ultra Reliable Low-Latency Communications (URLLC). eMBB is the natural evolution from the previous LTE cellular network where the capacity, the connectivity and the mobility are the cellular network attributes that are ex-

pected to significantly improve, exploiting for example, millimeter-wave communications [6]. Conversely, mMTC is expected to deal with the enormous amount of devices envisioned to be connected in the near future through the Internet of Things (IoT) paradigm, where the devices transmit relatively small amounts of data, need low bandwidth and are optimized to reduce their energy consumption. Lastly, URLLC is intended to address two orthogonal weaknesses faced in contemporary cellular networks: reliability and low-latency.

On the one hand, as Shannon proved in the information theory founding landmark paper [7], given a noisy discrete channel and a transmission rate smaller than the channel capacity, an encoding scheme capable of generating an equivocation rate ( $\epsilon$ ) arbitrarily small exists. This theoretical result shows how reliability depends on the coding scheme, and confirms that new coding schemes such as low-density parity-check (LDPC) codes can achieve arbitrarily small equivocation rates at the expense of using large data blocks [8]. 3GPP has already defined the new channel codings [9] and the successful achievement of reliable communications is indispensable for the URLLC adoption. On the other hand, a large amount of efforts is being invested by 3GPP trying to mitigate or eliminate the latency introduced by the 5G stack (e.g., new numerology for mini-slots or preemptive scheduling [10]) and the 5G procedures (e.g., uplink granted free transmission to avoid the Scheduling Request procedure delay [11]). This delay, as generated by the protocol stack itself can be considered as 5G endogenous. Moreover, the new SDAP sublayer [3] for classifying different services has been introduced, and a new QoS flow indicator or QFI has been defined to classify the different flows according to their different requirements [1]. However, even though these improvements undoubtedly reduce the latency, other aspects that reside outside of 5G's specifications, and that can be therefore classified as exogenous causes, may ultimately become the predominant factor in the latency. Examples of exogenous causes are the bufferbloat phenomenon or the RLC segmentation/reassembly procedure.

Contemporary wired technology largely surpasses wireless data forwarding speed (e.g., Gigabit Ethernet vs. LTE [12]), placing the bottleneck of the data path at the slowest data link of the chain, which is the the RAN. This observation remains true for newly deployed 5G networks [13]. Additionally, RANs are equipped with large buffers, aiming to avoid wasting scarce cellular network transmission opportunities, the capacity of which greatly varies due to the radio link dynamic nature. These two conditions of being the bottleneck and having large buffers, suf-

vice to mislead TCP Cubic [14], the most widely deployed TCP congestion control algorithm, causing packets from all flows to become queued for large periods of time in these oversized buffers.

In essence, and since packets are not lost but delayed, TCP Cubic's congestion control algorithm cannot differentiate the delay produced by a long path, and thus caused by the propagation delay, and the delay generated by bloated buffers, which is caused by large sojourn times at queues. Consequently, TCP Cubic misguidedly bloats the buffers, believing that a large propagation delay exists, and accumulating a large amount of packets at the bottleneck buffer. This scenario, however, does not present any problem if there exists only a single flow in the path. Packets from the single flow arrive to the buffer, wait a possibly large sojourn time, and are delivered as fast as the bottleneck's throughput permits it. In fact, at every transmission opportunity, a packet is waiting at the buffer, and thus, the throughput is maximized, and the latency for this lonely flow is minimized. However, in an scenario with multiple flows that share buffers, if one of the flows bloats the buffer, the packets belonging to the rest of the flows, will suffer large sojourn times caused by the flow that bloated the buffer. Such an effect, known as bufferbloat [15], would vanish if every flow would own a buffer for itself, as none of the flows would suffer delays generated by other flows. Unfortunately, the solution of creating a new buffer for each flow does not scale, since memory is limited, and creating a new buffer consumes time, which may be unacceptable in real time systems. Hence, such a solution cannot be considered for the general case, even though segregating and assigning a different queue to low-latency flows, reduces the latency suffered by the bufferbloat and should be used whenever possible. Furthermore, 3GPP describes a funnel scenario [1], where a maximum of 64 QFIs must be mapped to a maximum of 30 Data Radio Bearers (DRBs) [16], thus forcing flows with different requirements to share buffers along the cellular network stack. Moreover, in contemporary cellular networks, the pathological case where all the flows are mapped into one DRB is regrettably common, as the segregation of flows and the DRB generation must be agreed beforehand between the infrastructure provider and the service provider.

Thus, in buffer sharing scenarios, if one of the flows bloats the common buffer, the other flows' packets will have to wait for the buffer to deplete, in a phenomenon that can add a delay up to the order of seconds [15]. Such scenario is inevitable as packets belonging to a DRB should be

treated equally [1]<sup>1</sup>. This exogenous effect that is not a direct consequence of the 5G stack's architecture, can be critical for delay-sensitive applications such as Voice over IP (VoIP) or mobile gaming, that will share queues along the data path with other data flows, albeit having more stringent delay requirements.

Another important exogenous effect present in contemporary cellular network stack is the RLC's packet segmentation/reassembly procedure, that arises due to the packet-switch network nature of the Internet. Cellular network's RLC sublayer [5] [4] segments the current packet if the amount of bytes requested by the Media Access Control (MAC) [19] sublayer is smaller than the current packet size. Consequently, part of the packet is transmitted while the rest waits at the sender's RLC buffer until the next transmission opportunity arrives. The following transmission opportunity may arrive during the next TTI, in the best possible scenario, or may arrive after several TTIs, increasing the delay suffered by the packets, as the information in the receiver's RLC sublayer cannot be forwarded until the packet is completely formed. A maximum of 30 DRBs per User Equipment (UE) [16] is described by 3GPP, to which the MAC scheduler assigns resources according to different policies (e.g., Round Robin) and priorities. Hence, a naive MAC scheduler resource distribution that does not consider packets size's, leads to a plethora of fragmented packets enqueued in different RLC buffers, increasing the delay. Moreover, slicing will also emerge as a new 5G feature [1], challenging an efficient resource distribution [20], and thus, aggravating the segmentation/reassembly problem, since the segmentation/reassembly procedure, contributes to generate a non-negligible delay in the packets.

## 1.1 Objectives and Contributions

The main objective of this thesis is to propose different novel algorithms that reduce the latency in the cellular networks. Specifically, we have considered the exogenous delays that appear in the 5G cellular network stack and contribute to severely deteriorate its latency. Two main phenomena have been identified: the bufferbloat and the RLC segmentation/reassembly procedure. Both effects have been thoroughly investigated in this thesis, and efficient solutions to address them have been proposed. This has been achieved through the following contributions:

---

<sup>1</sup>In current open source LTE implementations [17] [18] the buffers are implemented as First In First Out FIFO queues.

- The suitability of existing low-latency bufferbloat algorithms in 5G's QoS architecture has been tested. We thoroughly studied 5G's QoS architecture, and we identified some of its weaknesses. Concretely, we identified the bufferbloat associated problem that arises due to 5G's QoS funnel nature. We confirm the conjectures induced through the careful 3GPP standard documents reading, in an emulated 5G QoS hierarchical multi-queue framework<sup>2</sup>, where we tested some of the most popular bufferbloat avoidance algorithms. In more detail, we tested the Active Queue Management (AQM) algorithm Controlled Delay (CoDel) [21], as well as Bottleneck Bandwidth and Round-trip time algorithm (BBR) [22] and Dynamic RLC (DynRLC) [23]. The results were published at the European Conference on Networks and Communications (EuCNC) (C1) and IEEE Transactions on Mobile Computing (J1).
- The design and evaluation of novel bufferbloat avoidance algorithms within 5G's QoS architecture has been performed. We carefully studied mechanisms like the Byte Queue Limit (BQL) [24] algorithm present at the Linux stack traffic control user-space utility program, where the amount of bytes at the queues are limited, and proposed the DRQL algorithm. We also approached to the problem through queuing theory, exploiting Kleinrock's work [25]. There, the optimal pacing rate at which a queuing system should work (i.e., not bloating the buffers, albeit achieving full throughput) is concisely described. Aiming to transmit packets at the optimal pace, we developed the (e)5G-BDP, as well as the USP considering 5G's specificities. We evaluated the proposed algorithms in the 5G QoS hierarchical queuing emulator, as well as in a tested using OpenAirInterface (OAI), and realistic mobility data [26]. The results were published in IEEE Transactions on Mobile Computing (J1) and IEEE Access (J2).
- A quantitative analysis of the latency generated by the RLC segmentation/reassembly procedure has been completed. The RLC segmentation/reassembly procedure segments part of the packet at the RLC sender's sublayer if the packet size exceeds the amount of bytes requested by the MAC. The segmented part of the packet waits at the receivers RLC sublayer until the rest of the packet arrives and a reassembly occurs. We quantitatively measured the delay generated by the segmentation/reassembly procedure in a real testbed.

---

<sup>2</sup>The code of the emulator can be downloaded from <https://github.com/mirazabal/Dynamic-buffer-TMC>.

The results were published in IEEE Access (J2).

- A design and evaluation of a RLC segmentation/reassembly procedure avoidance algorithm has been performed. Considering the cellular network specificities, we developed a scheduling algorithm named EQP, that elastically lends and borrows RBs, reducing the segmentation/reassembly procedure while achieving statistical fairness. EQP was tested in an enhanced OAI testbed considering a network slicing scenario with 2 COTS UEs under realistic traffic conditions [26] against a strict spectrum scheduling. The results were published in IEEE Access (J2).

## 1.2 Thesis Outline

Having the motivation and the contributions of this thesis described, the outline of this document is given in the following paragraphs.

In Chapter 2, the 5G QoS model is described and the entities/sublayers involved in our research are briefly detailed. The reasons why the bufferbloat appears and its consequences in the delay are later described. Following, some solutions that mitigate the bufferbloat are presented and, lastly, other possible 5G low-latency enablers are mentioned.

Chapter 3 elaborates the current 5G state-of-the-art bufferbloat avoidance mechanisms and studies the problem with 5G particularities. Next, three algorithms that emerged through the study of Linux traffic control and Kleinrock's work are presented: DRQL, (e)5G-BDP and USP. DRQL emerged while carefully studying the Linux traffic control mechanisms, while (e)5G-BDP and USP were discovered while modeling the cellular network through queuing theory. All the algorithms are evaluated in an emulator, as well as a testbed with COTS equipment.

On the other hand, Chapter 4 discusses one problem that arose while carefully studying the bufferbloat and that gets accentuated when the packets to transmit are large and the radio link capacity scarce. It first describes the RLC sublayer in detail and the segmentation/reassembly feature, present in the RLC Unacknowledged Mode (UM), as well as the RLC Acknowledge Mode (AM) mode. The effect of the segmentation/reassembly in the latency increase is described and the EQP algorithm to reduce its impact is presented. Following, the EQP is evaluated in a testbed considering MCS variations, slices, different traffic patterns and off-the-shelf equipment.

---

The conclusions of this thesis are summarized in Chapter 5, and possible future work regarding low-latency 5G mechanisms are lastly presented.





# CHAPTER 2

---

## Background: The 5G stack QoS model and the Bufferbloat

---

*In this chapter we first describe 3GPP's 5G stack and its QoS model. 5G's paradigm change from an LTE monolithic approach to the Service Based Architecture (SBA) is exposed, and the main functionalities are described. Following, we explain the bufferbloat phenomenon and we present current solutions that rely on different approaches. This chapter ends describing some other low-latency enablers that do not directly address the bufferbloat problem, while still relevant for reducing current latency in future cellular networks.*

### 2.1 5G Stack and its QoS Model

The ambitious objective of 3GPP of creating a standard capable of successfully dealing with a plethora of different traffic flow requirements, requires a complex QoS handling that is precisely

described in [1]. Additionally, QoS has received significant attention in the recent years from the research community in the wired (e.g., Institute of Electrical and Electronics Engineers (IEEE) 802.3 [27] or IEEE 802.1Q [28]), and wireless (e.g., IEEE 802.11 or cellular networks) technologies [29] [30].

In 5G, 3GPP decided to decouple the cellular network functionality among different entities at the Core Network (CN), changing its paradigm from a monolithic to a microservice approach, aiming to reduce the current service providers' dependency with vendors. The current 5G model for the CN is based on a SBA as seen in Fig. 2.1. It consist of:

- The Network Slice Selection Function (NSSF), which redirects the traffic to a network slice.
- An Authentication Server Function (AUSF), which provides security in UEs authentication.
- An Access and Mobility Management Function (AMF), which is responsible for addressing connection and mobility management tasks.
- An Application Function (AF), which provides session related information to the PCF.
- A Data Network (DN), which represents the network from which data is retrieved and sent. For most of the cases, we can think of the DN as the Internet.
- The User Plane Function (UPF), which acts as the connection point between the DN and the RAN. It routes, filters, detects and forwards packets, reports traffic usage and handles packets per QoS flow. It is the first entity in the downlink procedure, where traffic engineering techniques can be applied, as well as the first buffers that a packet will visit in its way to the receiver, as seen in Fig. 2.3.
- The Policy Control Function (PCF), which provides policy rules to govern network behavior (e.g., charging for a service data flow).
- A Session Management Function (SMF), is responsible for, among other functions, the UE Internet Protocol (IP) address management, the control part of policy enforcement and QoS, and managing the Packet Control Protocol (PCP) session.
- A Network Exposure Function (NEF), which provides a unified method to provide data

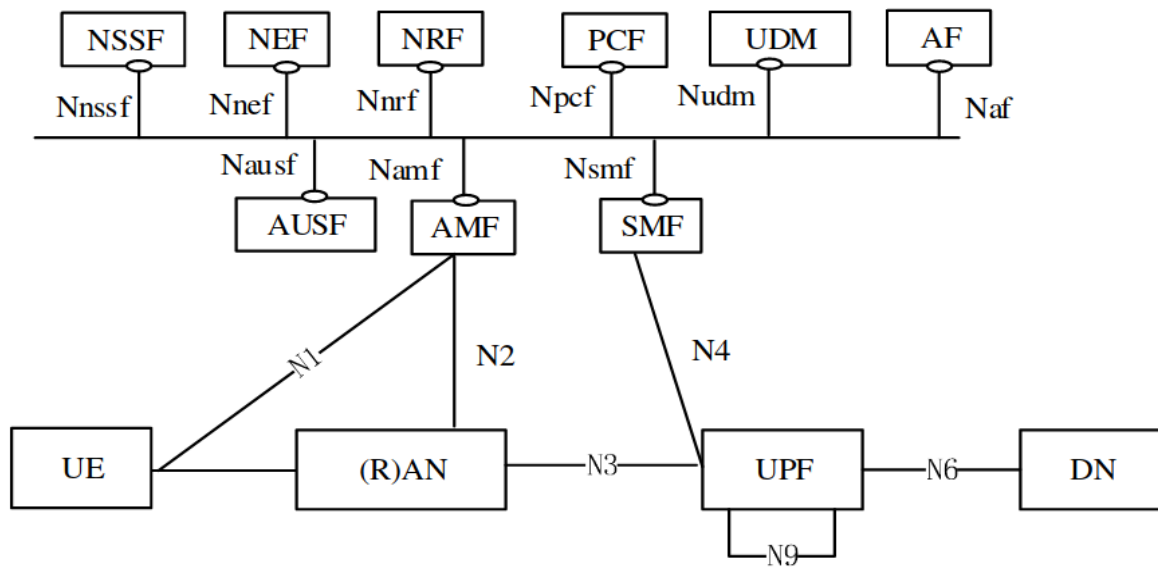


Figure 2.1: 3GPP SBA. From 3GPP Technical Specification (TS) 23.501 [1].

to external services.

- A Network Repository Function (NRF), which mainly maintains the profiles of the available network function instances and their support.
- The Unified Data Management (UDM), which provides similar functionality as the Home Subscriber Server (HSS) from LTE, which provides subscribers information to other entities within the network.

Regarding the RAN, two layers have been standardized: the Layer 1 that describes the Physical Layer PHY, and the Layer 2 that describes MAC, RLC, Packet Data Convergence Protocol (PDCP) and SDAP sublayers, as shown in Fig. 2.2. These entities have been defined with the aim of bringing flexibility, as well as improving decoupling and encouraging compatibility between different vendors. Lastly, the Radio Resource Control (RRC) [16] is responsible for the radio bearer establishment, reconfiguration and release, mobility procedures, paging notification or broadcasting system information.

To explain the route that a packet takes and the main QoS mechanisms that encounters while traveling, in the following, the key aspects of such 5G QoS scenario for the downlink are presented, while a similar approach applies to the uplink due to 5G's symmetry. Packets arrive from the DN through the N6 interface to the UPF (cf. Fig 2.1, 2.3 and 2.4), where the first buffer

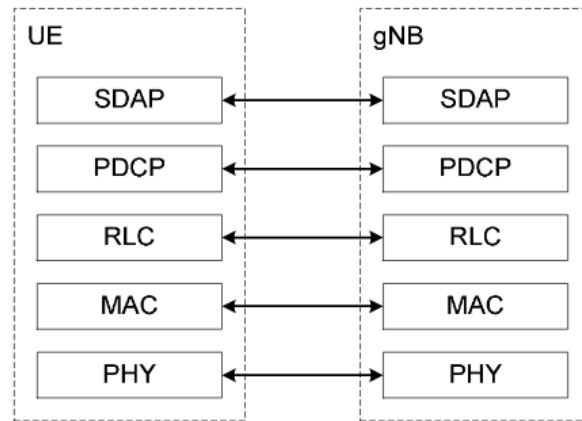


Figure 2.2: 5G's RAN stack. From 3GPP TS 38.300 [2].

that a data packet will encounter in its path to the UE is located. The UPF identifies and segregates these data flows based on the configuration received from the SMF. PCP session describes how packets should be identified and marked with its QFI through the Packet Detection Rule (PDR) (cf. Fig. 2.3 and 2.4), policed through the Multi-Access Rule (MAR), forwarded based on the Forwarding Action Rules (FARs), tagged based on the QoS Enforcement Rules (QERs) and lastly reported using the Usage Reporting Rules (URRs) [1] [31]. UPF is the first entity in the 5G downlink scenario where traffic engineering techniques and packet buffering takes place as depicted in Fig. 2.4. Packets egressed from the UPF are already marked with a QFI as observed in Fig. 2.3 and 2.4.

QFI is a 6 bit field (i.e.,  $2^6 = 64$  possible priorities). Every QFI is associated with a set of characteristics according to [1] among which, the maximum data burst, the resource type, the priority level, the Packet Delay Budget (PDB) or the Packet Error Rate (PER). Three different resource types are described: Guaranteed Bit Rate (GBR), Non-Guaranteed Bit Rate (Non-GBR) and Delay-Critical GBR [1]. The priority level indicates the importance of the packet for scheduling purposes. The PDB is the upper bound of the delay permitted, measured from the N6 interface until the UE [1]. It is also considered to determine the scheduling weight and the Hybrid Automatic Repeat Request (HARQ) retransmissions [1]. PDB only considers the delay within the 5G network stack, and thus, it does not consider the end-to-end delay of the user application. The PER is defined as the number of packets processed by the RLC sublayer of the sender, divided by the number of packets delivered to the PDCP sublayer at the receiver.

According to [1], a single UPF or multiple UPFs can be provided for a given Packet Data Unit

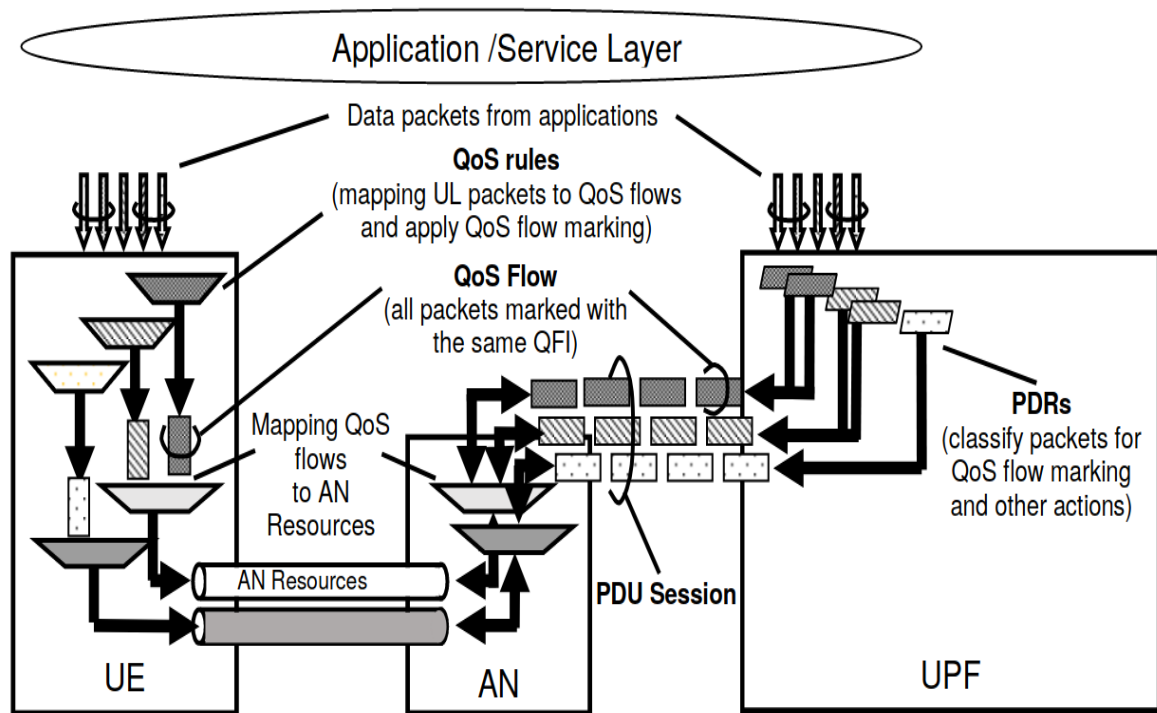


Figure 2.3: 5G's QoS scenario. From 3GPP TS 23.501 [1].

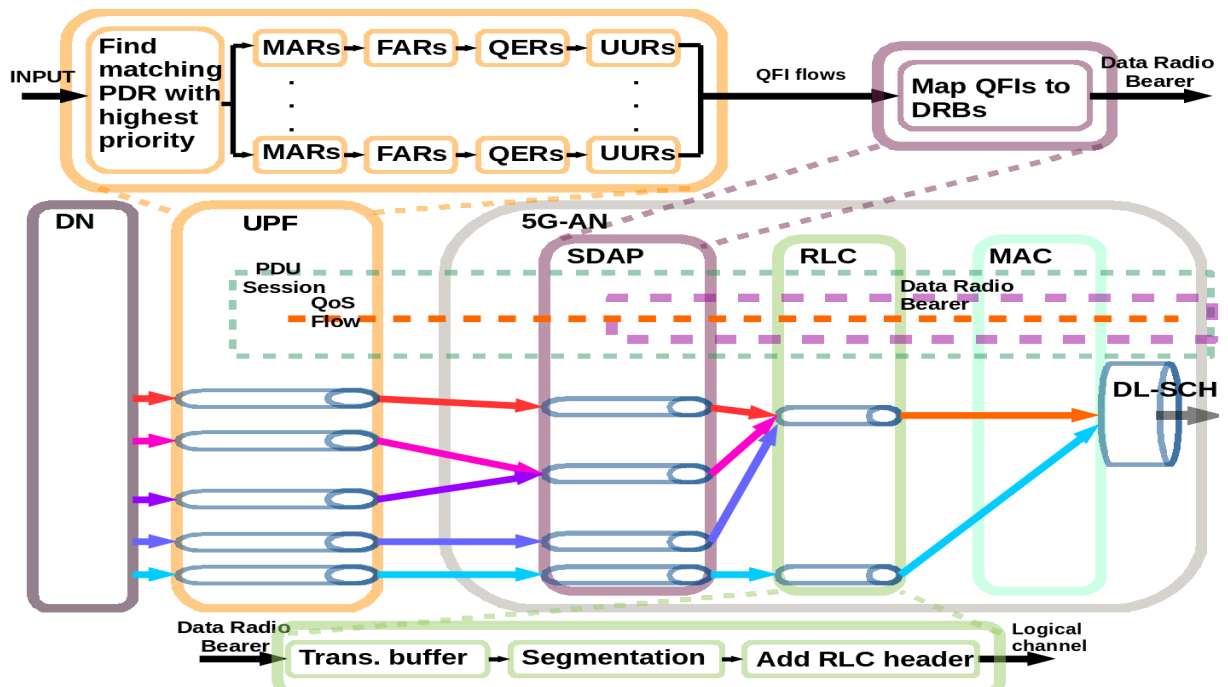


Figure 2.4: Simplified 5G QoS downlink block diagram with the entities, buffers, QoS flow and DRB abstractions.

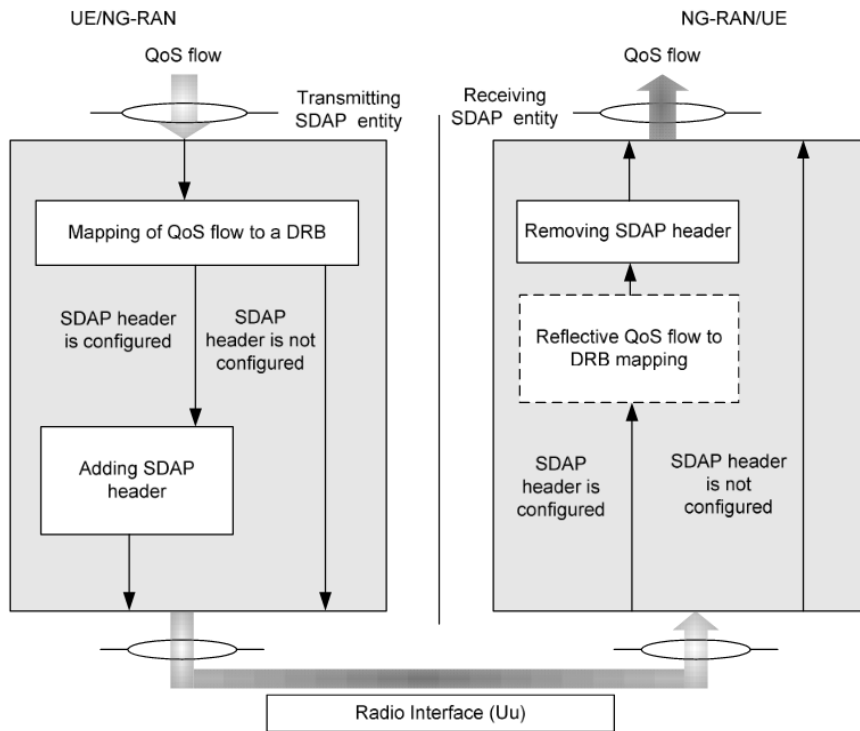


Figure 2.5: SDAP functional view. From 3GPP 37.324 [3].

(PDU) session. The UPF selection is decided by the SMF. If deterministic low-latency is required, the UPF will be created as close as possible to the RAN. Software Defined Networks (SDN) will provide the possibility of instantiating and scaling entities dynamically, therefore enabling the relocation of UPFs, and thus reducing the latency, as demonstrated in [32] and described in [33].

Packets will then be forwarded through the General Packet Radio Service (GPRS) Tunneling Protocol (GTP) to the SDAP sublayer at the RAN [3] (cf. Fig. 2.4), where the second buffer that a data packet will encounter in its path to the UE is placed. A SDAP per PDU session is described in [3], even though it is mentioned that other implementations are possible. RRC configures this sublayer to map the QFIs assigned by UPF into DRBs [16] as seen in Fig. 2.4. A maximum of 64 QFIs and 30 DRBs are allowed per UE [16]. Therefore, many to one mapping needs to be defined at the SDAP sublayer and data packets marked with different QFIs will inevitably share queues following the pigeon-hole principle. 3GPP defines the SDAP *raison d'être* as a simple mapper as shown in Fig. 2.5.

This implies that a FIFO structure is foreseen since no scheduling capacity is enabled. Packets

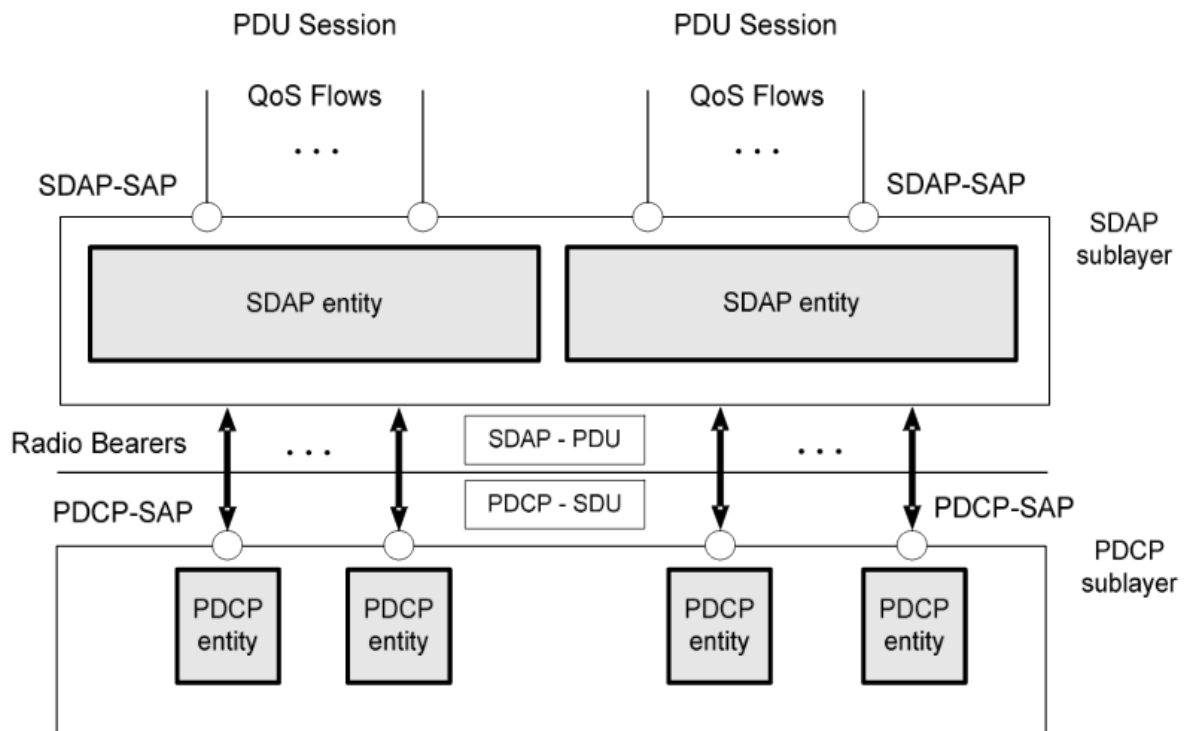


Figure 2.6: SDAP structural view. From 3GPP 37.324 [3].

are then passed to the PDCP [34] sublayer, as depicted in Fig. 2.6. PDCP is responsible for header compression, ciphering and integrity protection among other things. No buffering occurs at PDCP unless a packet reordering to upper layers is configured, as for example due to dual connectivity. Lastly, the data packets arrive to the RLC sublayer, where they are buffered, segmented and the RLC header is added in the transmission side. A RLC entity per UE and DRB will be instantiated. This is the last queue described by 3GPP since MAC is not provided with one, and we can neglect the HARQ queue since it does not consume data but rather retransmits it. 5G's Layer 2 sublayers and mappings between sublayers in downlink can also be seen at Fig. 2.7. Packets wait at the RLC until a MAC sublayer notification arrives to forward the packets.

Fig. 2.7 shows an overall 3GPP picture of the Layer 2 sublayers downlink procedure, the mapping from QFI to DRB at SDAP, the buffering at RLC, and the scheduling capabilities at the MAC.

However, if heterogeneous traffic delay and priority constraints must be met, mobile network operators will go beyond 3GPP, and deliver a packet scheduling solution at the SDAP sublayer

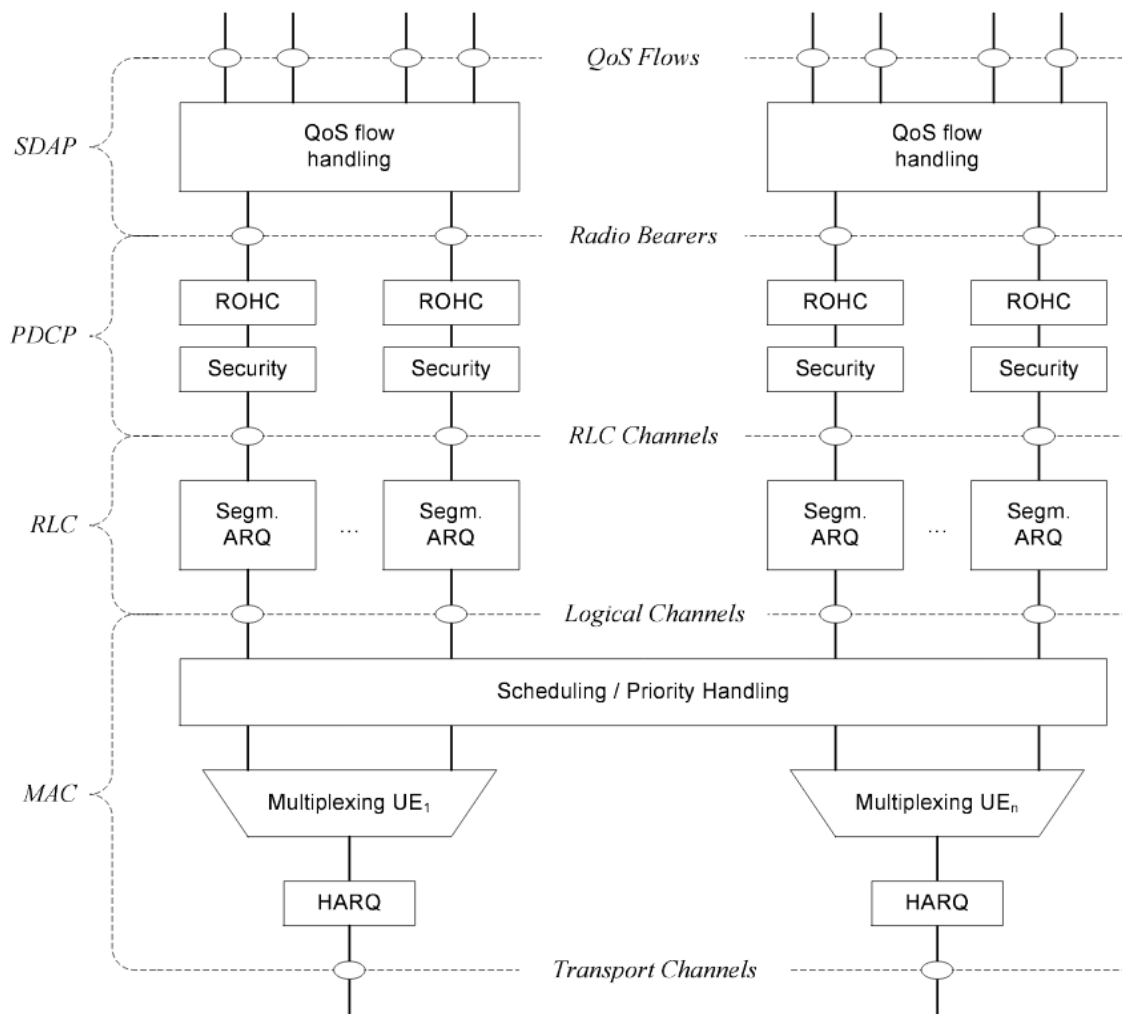


Figure 2.7: 5G's Layer 2 SDAP, PDCP, RLC and MAC sublayers in downlink procedure. From 3GPP TS 38.300 [2].

that can selectively forward packets with different requirements, since once a packet is assigned to a queue, segregating it according to their QFI can be costly and complex. In essence, they will provide a backlog packet mechanism to avoid the delays generated by the bufferbloat phenomenon. Moreover, if a scheduler is not implemented at the SDAP sublayer, the only possibilities for traffic engineering where buffering occurs is the UPF [31] and the RLC [4] as illustrated in Fig. 2.4. Since the UPF network function and the RLC sublayer are not contiguous, if a backlog mechanism that avoids the bufferbloat is required, the delay for communicating both entities may be unacceptable in 5G low-latency communications.



## 2.2 Bufferbloat Solutions and Delay-Sensitive Enablers

The term bufferbloat was coined to name the latency increase originated by the presence of excessive large (bloated) buffers in systems [35], and it is also a concern in other technologies that rely on QoS prioritization mechanisms (e.g., IEEE 802.1Q Virtual Local Area Network (VLAN) [28]). In the presence of large buffers in the bottleneck data link, TCP's congestion control algorithm (e.g., TCP Cubic) capability of estimating the available bandwidth gets distorted, and therefore, increases its sending rate until a packet is dropped, leading to a plethora of packets at the slowest link's buffer, a.k.a. the bottleneck, during the process, interpreting the generated buffer sojourn time at the bottleneck as the data path's distance. In essence, the congestion control algorithm cannot differentiate between the packet's propagation time and the sojourn time that a packet suffers at bloated buffers. To overcome it, different approaches have been developed on the wired and IEEE 802.11 domains at various levels (e.g., AQM algorithms, TCP Small Queues (TSQ) [36], TCP Segmentation Offload (TSO), BQL [24]), or new congestion control algorithms have been proposed, such as BBR [22], and more recently Cellular Controlled delay TCP (C2TCP) [37]. In the following, we explain the most notable examples in each field.

To avoid the bufferbloat effect AQM solutions were developed. AQM was designed to maintain the buffers within a reasonable size, and thus avoiding the bufferbloat, while achieving the maximum possible throughput. When an AQM queue starts getting overloaded, it discards packets as a measure to notify the sender that the transmission rate should be reduced. Therefore, the queue does not get bloated, impeding the link saturation of greedy flows, and, thus, if another flow shares the same queue, large sojourn times are avoided. However, discarding the packets at the appropriate moment to avoid the bufferbloat, while utilizing all the transmission opportunities in an environment where the throughput dynamically changes and the packet arrival sequence is unknown beforehand, is very challenging [15].

Random Early Detection (RED) [38] appeared as the AQM algorithms' pioneer. Even though the first results were very promising, it was never widely implemented in consumer network routers. RED considers the growing rate of the queue as a congestion symptom, and increases the probability of discarding a packet accordingly. While persistent queues indicate congestion, the growing rate of a queue does not, as many TCP implementations have a default bulky behavior (e.g., TCP Cubic). The bursty traffic nature of concurrent TCP sources can grow and shrink

the queues before RED can effectively react accordingly [39]. Hence, the RED algorithm was never widely adopted in consumer electronics. Following, BLUE [39], took a fundamentally different approach to manage congestion, considering lost transmission opportunities or link idle events and packet losses. The probability of dropping the next packet is increased if a packet lost due to a buffer overflow occurred and, on the contrary, the probability of dropping a packet is reduced if a transmission opportunity loss happens. The Adaptive Virtual Queue (AVQ) [40] algorithm appeared later, where the arrival packet rate is the governing factor. A parallel virtual queue per real buffer is instantiated, and according to the current virtual capacity of the queues, the decision to discard a packet (or mark it through the Explicit Congestion Notification (ECN) flag) or egress it is reached. The virtual queue capacity is dynamically adapted according to the arrival packet rate. The rationale behind this idea is that marking should be more proactive if the arrival rate exceeds the egress link rate. The probable idea not mentioned in the article is the Bandwidth Delay Product (BDP), which points the optimal pacing rate for a queuing system [25], maintaining the arrival rate equal to the maximum possible egress rate, even in dynamic scenarios.

CoDel [41] is a newer AQM algorithm that aims to improve RED and BLUE's shortcomings. CoDel is a packet sojourn time based algorithm that discards packets to inform the sender's TCP congestion control algorithm that excessive buffering is taking place. It is one of the best known AQM solutions, and a widely implemented solution to address the bufferbloat problem [42]. It is governed by two variables, the *interval time* and the *target time*. CoDel inserts a timestamp in every packet that is enqueued. During the *interval time*, the sojourn time of every egressed packet is measured, and the minimum value is saved. If after the *interval time*, the minimum saved value is above the *target time*, the next packet is dropped as a mechanism to inform TCP's sender flow that excessive queuing is happening. The *interval time* is then decreased by  $1/\sqrt{x}$ , where  $x$  starts at 2 and increases every interval where the desired sojourn time is not reached. The inverse of the square root is selected since the drop rate in the network is inversely proportional to the square of the throughput in a loss-based TCP congestion control algorithm, as demonstrated in [43]. The *interval time* is recommended to be set to 100 ms, while the *target time* is recommended to be a 5% of the *interval time*. It is classified as a knob-less AQM solution as no parameters have to be tuned, since the recommended values are considered valid for nearly all the possible scenarios. All these algorithms try to inform end-points about the congestion so that they can act accordingly.

After CoDel's appearance, the Proportional Integral controller Enhanced (PIE) [44] was proposed by Cisco. PIE is composed of a random drop that happens before a packet is enqueued, a drop probability calculation block and a latency calculation block. According to the latency calculation, the drop probability is updated, which directly affects the random drop block. In fact, it is a very similar model to RED, with a focus on latency rather than occupancy. It aims to reduce CoDel's overhead avoiding creating a timestamp per ingressed packet. However, some recent studies [45] argue that estimating the latency through the departure rate may not be a valid parameter and recommend using a timestamp at packet ingress.

Another important mechanism to manage the low-latency requirements in the bufferbloat context present in contemporary literature is the limitation of the queue sizes, keeping the packets in the upper layers. Since packets that are aggregated into one queue are treated equally, maintaining the queues as empty as possible enables the possibility to avoid large latencies associated with large queue sojourn times. This enables delay-sensitive packets to evade large queues created by other flows. In recent years, this principle has been consistently applied at different levels in the Linux kernel's TCP/IP stack with great success. One example is the TSQ [36], where the size of the queues are reduced to avoid intermediate buffering with partial success. However, as [46] demonstrated, limiting the queues excessively may starve the data link, and therefore, a balance is to be found. Another example is the BQL [24], which relies on the fact that the last queue before the PHY layer, should be provided with enough bytes but not more. It also shows the problems associated with handlers rather than with bytes, and it advises to deactivate the TSO. TSO is an optimization from the kernel that reduces the CPU cycles and permits the kernel to send packets bigger than the maximum Maximum Transmission Unit (MTU) to the Network Interface Controller (NIC), where the NIC driver is responsible for segmenting such packets before sending them through the link. Such an optimization disables the possibility of delivering low-latency packets since once a large packet gets into the hardware, the low-latency constraint packet will suffer the sojourn time associated with the depletion time of the larger packet.

Scheduling is another important pillar when it comes to QoS and the bufferbloat since higher priority traffic should access the resources first. This is a crucial aspect for low-latency delay-sensitive traffic. In any case, a balance has to be reached to avoid starving some flows, while prioritizing others, especially for Non-GBR traffic. One of the first network algorithms that

addressed such a problem is the Stochastic Fair Queuing (SFQ) [47]. Flows are hashed and assigned to different queues. Since in principle, every flow has a different queue the bufferbloat effect disappears as every flow is just responsible for the self generated delay. Every active queue is assigned an equal egress rate in a Round Robin manner. However, due to the hashing collision possibility, two flows can end sharing a queue, splitting each flow's theoretical share of bandwidth and suffering queue sojourn times generated by other flows. This situation is partly alleviated by periodically adding a perturbing value to the hash function that rehashes the flows, thus reducing the possibility of different flows sharing the same queues for large periods. However, due to 5G's funnel QoS architecture described in Section 2.1, buffer sharing between different flows will become unavoidable. One improvement over SFQ is the Deficit Round Robin (DRR) [48]. In SFQ, different flows might have different packet sizes and therefore, the fairness is packetwise but not bitwise. Traffic sources that send packets with smaller size, would get less than its corresponding bitwise bandwidth. DRR adds a quantum value that measures how much bandwidth corresponds to each active queue. If the packet at the queue is smaller than the quantum value, the packet is subtracted and the quantum value is reduced by the packet size. If, on the contrary, the packet size surpasses the quantum size, the quantum value is accumulated for the next round. In this way, a bitwise fairness is assured. One more modern approach is the DRR++. In this scheduler, the latency sensitive traffic demands are handled. The sender agrees to send less than one quantum during a round of the scheduler. As long as the sender does not surpass this rate, the scheduler guarantees that only higher priority traffic will delay this flow. If the sender surpasses this rate, these new packets will not be considered for the current round. In this way, the high priority traffic is not lost, even if the traffic is transported through a bursty protocol. CoDel, in conjunction with a DRR scheduler [48], forms the FlowQueue-CoDel Packet Scheduler (FQ-CoDel) [49]. FQ-CoDel is nowadays the most successful low-latency algorithm, activated by default at some embedded open source router projects [42]. The idea of scheduling is closely related to the idea of segregating the flows in different queues. However, the bufferbloat only appears if one of the flows presents greedy behavior. Therefore, in Low Latency, Low Loss, Scalable Throughput (L4S) [50] authors propose segregating the traffic prone to generating the bufferbloat from the traffic that is not. Such an approach requires more resources as a new queue has to be generated, but considerably reduces the sojourn time of non-bulky flows. It is also worth mentioning that the QoS and scheduling problem is a major problem in IP packets' networks, which introduced the

Differentiated Services (DiffServ) architecture to handle different priorities and requirements<sup>1</sup>. DiffServ is a traffic management mechanism that handles different QoS flows according to the classes that they belong (e.g., giving preferential treatment for higher-priority flows).

The bufferbloat problem can also be tackled from the TCP's congestion control algorithm point of view. Kleinrock proved in 1979 that there exists an optimal rate at which the packets from a flow should be transmitted [52] and, unfortunately shortly after, Jaffe proved that such a metric could not be achieved through a decentralized algorithm [53]. Thus, the idea of transmitting packets at an optimal rate was partially abandoned due to the Internet's decentralized nature in favor of loss-based algorithms (e.g., TCP Cubic). However, based on the results achieved in [52], Google presented a novel congestion control algorithm (i.e., BBR) [22]. It calculates the Round Trip propagation (RTprop) delay and the Bottleneck Bandwidth (BtlBw) from every acknowledgment (ACK) through a sliding window. If an increase in the measured RTprop is detected, which is estimated through the Round Trip Time (RTT), BBR translates it as a congestion symptom and acts accordingly reducing the pace of forwarding packets. Every 10 seconds<sup>2</sup>, BBR empties the generated queues and recalculates the optimal throughput value. BBR, contrary to Cubic, is not directly affected by packet losses, and therefore, it appears as a more strong candidate for mobile networks. Once BBR calculates the BtlBw through TCP's ACKs elapsed time, it delivers the packets in an appropriate pace, and thus, the packet accumulation at the bottleneck buffers is avoided.

## 2.3 Other 5G Low-Latency Enablers

Slicing is envisioned as a fundamental technology to solve URLLC traffic flows requirements. It relies on the old idea of virtualization that was developed in the late 60s by IBM. Each slice has some isolation properties, some particular QoS features and is an abstraction from the hardware, where the 5G stack runs. According to 3GPP [1], a slice is a *logical network that provides specific network capabilities and network characteristics*. However, the bufferbloat problem still arises if two flows with different requirements are assigned to the same queue in the same slice. In [54], the problem generated by queue depletion time when different slices with different traffic patterns

---

<sup>1</sup>The QFI in 5G is composed of six bits, which exactly matches the bits needed by DiffServ, resulting the mapping from DiffServ to QFI trivial[51].

<sup>2</sup>According to the implementation from the Linux kernel version 5.3.0-62-low-latency.

(e.g., eMBB, mMTC and URLLC) that compete to access scarce resources is analyzed, and the shortcomings from slicing are implicitly shown. Additionally, segregating contemporary traffic flows into different slices presumes that information about the packets and its characteristics are shared with the cellular network *a priori*. This is a weak assumption in contemporary societies where privacy is becoming a concern, and therefore, encrypted data and relay packets will become more common. Thus, the deep packet inspectors capacity to segregate the packets present at cellular networks will considerably be reduced.

Another approach that reduces the delay in 5G is the possibility of reducing the physical distance to the servers through Mobile Edge Computing (MEC). This solution is orthogonal to the bufferbloat phenomenon as the former aims to reduce the time to retrieve the information (i.e., it deals with the propagation delays associated with the DN). MEC servers reside outside the UPF and may be accessed directly from the UPF network entity instead of forcing the query to access the Internet with the associated delays (i.e., it will act as a cache memory). It also gives the possibility of delegating CPU power from the UE to the cloud, as Gaming as a Service (GaaS) envisioned [55]. The two described capabilities (i.e., using it as cache memory or delegating CPU power) deserve a different approach as they represent completely different semantics even though they share the same word (i.e., MEC). 3GPP is also investing effort in finding the best approach for different use cases [33]. Another recent study [56], applies the idea of utilizing the information from the cellular network to fine tune MEC's TCP algorithm. Even though promising results arrive with similar bandwidth reduction as CoDel, MEC solutions will be out of the scope of the solutions that cannot afford more than milliseconds delays, due to the latency to transmit the network congestion information between the RAN and the MEC.

In summary, slicing and MEC are large topics in cellular networks that can serve to reduce the latency, each deserving a profound study. However, both phenomena are orthogonal to the bufferbloat problem, and thus, are not thoroughly analyzed in this thesis.

## 2.4 Summary

In this Chapter 2, 5G's QoS stack with its new entities have been briefly described. The route that a packet traverses in the downlink procedure has been thoroughly described and 5G's QoS funnel nature has been exposed. Additionally, the bufferbloat problem has been shown, and

---

its effect in low-latency flows has been presented. It has also been described how the current 5G stack meets the required conditions (i.e., being the slowest link in the path and having large buffers to avoid squandering wireless resources) to generate large sojourn delays at its RLC buffers. Following, bufferbloat solutions that are nowadays used at different layers have been presented (i.e., AQM, schedulers and TCP's congestion control algorithms). Lastly two contemporary cellular networks low-latency enablers, slicing and MEC, have been exposed and their orthogonality with the bufferbloat explained.





## CHAPTER 3

---

# Addressing the bufferbloat phenomenon in 5G

---

*In this chapter we first present the state-of-the-art solutions with cellular network's specificities. We then describe how we mapped Linux traffic control solutions into the 5G network. Following, we present a study of queuing theory in 5G, and we describe the novel latency reduction algorithms that emerged from this study. We later present our evaluation frameworks: an emulator and a testbed. We lastly comment on the experimental results of the proposed algorithms, as well as, that of three other state-of-the-art solutions.*

### **Contributions:**

[J1] M. Irazabal, E. Lopez-Aguilera, I. Demirkol, and N. Nikaein, "Dynamic Buffer Sizing and Pacing as Enablers of 5G Low-Latency Services", (Early access), IEEE Transactions on Mobile Computing, 2020. (Area: Telecommunications; Quartile Q1; IF: 5.112).

[J2] M. Irazabal, E. Lopez-Aguilera, I. Demirkol, R. Schmidt and N. Nikaein, "Preventing RLC Buffer Sojourn Times in 5G", IEEE Access, March 2021. (Area: Telecommunications; Quartile

Q1; IF: 3.745).

[C1] M. Irazabal, E. Lopez-Aguilera, and I. Demirkol, "Active Queue Management as Quality of Service Enabler for 5G Networks," in 2019 European Conference on Networks and Communication (EuCNC), April 2019, pp. 1–5.

### 3.1 State of the Art in 5G Active Queue Management and Bufferbloat Avoidance Solutions

As exposed in Chapter 2, 3GPP's described 5G stack lacks a bufferbloat avoidance mechanism. Therefore, delays in the order of seconds [15] will be generated if no countermeasures are taken. In the following, the state-of-the-art mechanisms that appeared in recent literature are presented.

Even though a growing community around the bufferbloat problem [15] exists, and a consensus around its negative impact and importance in low-latency traffic prevails, no substantial effort has been invested studying the bufferbloat phenomenon within cellular networks specificities. The state-of-the-art delay-sensitive solutions in 5G, have almost exclusively focused on endogenous delays generated by the cellular network stack itself [57]. While such a field deserves a profound study, the end-to-end delay may be governed by 5G exogenous factors. In fact, in today's LTE networks, the physical time for transmitting a bit of information is inferior to 1 *ms*, if no transmission failure happens, which under all perspectives is a remarkable result<sup>1</sup>. In an evolved Node B (eNodeB) with a 20 MHz bandwidth or 100 RBs, up to 66 Kbits of information can be transmitted in one TTI that lasts for 1 *ms* in LTE. However and unfortunately, the transmission of information suffers from other delays, not directly associated with the physical layer, that substantially increase the total delay. One such phenomenon is the large sojourn times that packets suffer at bloated buffers, a.k.a bufferbloat. AQM was developed specifically to reduce the number of packets at bloated buffers, and thus, mitigate its corresponding sojourn delay.

One of the first wired AQM algorithms discovered was RED [38] and therefore, one of the first to appear in the cellular network context. A modified RED algorithm was evaluated at the RLC

---

<sup>1</sup>It is worth mentioning that just reading randomly 4KB of data from a SSD of 1GB/sec takes 150  $\mu$ s [58] in modern hardware, which makes the objective of 1 *ms* very challenging with current technology.

sublayer [59], which reported favorable results in a controlled environment. As mentioned in Chapter 2, its tedious configuration stopped its more widespread adoption in wired, as well as in wireless technologies. Other promising results have flourished recently, where a CoDel queue is placed at the RLC queue [60]. Unfortunately, the results obtained at [60] have not been reported to a real cellular network testbed, where problems related to the dynamic nature of the radio link may appear, and thus, its results should be carefully considered.

The SFQ on the 5G context has been explored in [23], where the authors proposed an algorithm named DynRLC, with a SFQ entity placed before the PDCP sublayer, and hence, segregating the traffic. In order for this technique to succeed, the RLC buffer size has to be limited to avoid bloating it, maintaining the packets at the PDCP sublayer. This idea is also presented in the same paper [23], where the authors describe an algorithm that shrinks or augments the RLC buffer capacity according to the RLC's Service Data Units (SDUs) packets sojourn time. This technique manages to maintain the RLC buffer considerably uncongested when compared with the default deployment. The same authors proposed the Enhanced Bearer Buffer [61], where the same principle is applied (i.e., measuring the RLC SDUs sojourn time and limiting RLC buffer capacity) within a finer interval time, which achieves better results, as it approaches to a pacing mechanism (i.e., packets are not submitted in a bulky manner) and avoids large sojourn times.

Some industrial approaches for 5G non-Standalone scenarios have foreseen a scenario where resources can be guaranteed by statically reserving bandwidth for them [62]. Since high priority flows will not yield the resources even if they are not using them, this approach leads to underutilization and lacks scalability.

Other more interesting approaches have been presented in [63], where an implementation of FQ-CoDel is described. In [63], the RTT of the packets is measured, the bandwidth is calculated, and the packets are sent slightly slower than the maximum estimated bandwidth. With such an approach, the link bottleneck is artificially created at the UPF, where a pacer and a scheduler select the rate and the packet to forward, respectively. Unfortunately, no empirical results are delivered. In any case, such an approach presents some important deficits. In the first place, the dynamicity of the 5G network can make the bandwidth change abruptly. In order for this algorithm to realize that the bandwidth has changed, some delay is expected as the information has to flow from the RAN until the UPF entity, which may lay at different geographical locations.

Moreover, while most applications that require reliable, ordered and error free data delivery are transported through TCP that generates an ACK packet from which the RTT can be measured, some services rely on other transport protocols for its data delivery (e.g., QUIC [64], Stream Control Transmission Protocol (SCTP) [65]), and thus, the suggested solution is infeasible for such scenarios.

Another important mechanism to manage the low-latency requirements explored in the literature is the limitation of the queues. In this case, packets are maintained segregated at higher queues instead of forming the queue at the bottleneck. Since packets that are aggregated into one queue are treated equally, maintaining the queues as empty as possible enables the possibility to avoid large latencies associated with large queue sojourn times. This allows the delay-sensitive packets to evade large queues. In recent years, this principle has been consistently applied at different levels in the Linux kernel's TCP/IP stack with great success in [36], [24], as well as in the cellular network, as presented in [66], [67] and in [68]. In [66], the traffic control mechanisms provided by the Linux kernel stack applied to the cellular context are analyzed, and the fact that the popular mobile Android Operating System is based on the Linux kernel is exploited, to propose a realistic solution using the BQL algorithm [24]. The BQL algorithm resizes the queue to its optimal byte size according to the last egress rate. Such a dynamical mechanism has proved to be adequate in contrast with the static small queues approach, which may reject packets before achieving full throughput [46]. Even though good simulation results were obtained, the study considers the cellular access network as a queue, while in reality, the QoS queuing is composed of multiple hierarchical queues [1] as shown in Fig. 2.4. Moreover, all the traffic inside the access network is treated equally, which deprives the possibility of a finer grained segregation and a more refined QoS guarantee.

In [67], a *Q-learning* algorithm for limiting the IEEE 802.11 MAC queues is presented under the name of *LearnQueue*. While very interesting results are obtained through a backlog mechanism, such design is not possible at the 5G MAC sublayer due to the lack of queues in 5G at the MAC sublayer. Additionally, a reinforcement learning method, even though it can theoretically maximize the total reward, it needs infinite exploration time for optimal results. If a new actor with unforeseen characteristics joins in, the algorithm needs some time until it can correctly schedule the resources. In a heterogeneous scenario as 5G, this fact could translate into a fatal transition time until the service can correctly be served.

Another interesting study about the bufferbloat in wireless technologies can be found in [29], where a solution for the accumulation at lower queues of the MAC sublayer is presented. Basically, if no backlog mechanism before the last queue occurs, the mechanisms that take place at upper layers lack effectiveness. Therefore, and taking into account the IEEE 802.11 particularities, the authors developed another level of queues that avoids the packet accumulation at the last queue. Their proposal has been integrated in recent Linux kernel versions.

Lastly, some efforts have been invested trying to address the bufferbloat problem through TCP's congestion control algorithm in cellular networks. In [37], the C2TCP is presented. There, a modified Cubic congestion control algorithm is used along with a sliding window in order to absorb the dynamic nature of the radio link. The congestion is detected through the RTT measurement, and it acts accordingly before a packet is lost as in the classic Cubic scheme. However, in [37] a mere 78% of the total bandwidth in comparison with Cubic is reported, which implies a non-negligible trade-off between bandwidth and latency. Other TCP approaches such as [30], where the authors present an algorithm named ABC, rely on sharing information between the RAN and the corresponding server, marking packets with a break or accelerate command depending on the congestion state of the cellular network. In any case, such TCP based approaches add a considerable delay, as the congestion information has to travel all the path until the sender, adding milliseconds order response delays. Such granularity may not be acceptable for delay-sensitive services that may not tolerate milliseconds order delay.

### **3.2 Linux Traffic Control Approaches in the 5G Stack: DRQL**

As the wireless domain is significantly slower than the wired domain in contemporary networks, the bottleneck, and therefore the bufferbloat, in actual cellular network systems resides at the entity that holds the last buffer in the wired domain (i.e., RLC sublayer's buffer). Thus, a solution for tackling the bufferbloat problem in 5G must inevitably start reducing the buffers' occupancy at RLC. A naive solution at RLC sublayer would assign a slightly superior buffer size per DRB than the maximum possible delivery rate to every UE under the best radio conditions, and if it is surpassed, packets will be dropped. However, such an approach creates big sojourn times at the RLC buffers if the MAC scheduler assigns a partial amount of available RB to the UE (since multiple DRBs compete for the available bandwidth), or if suboptimal radio channel conditions occur. Additionally, the foreseen 5G softwarization capability will be another factor

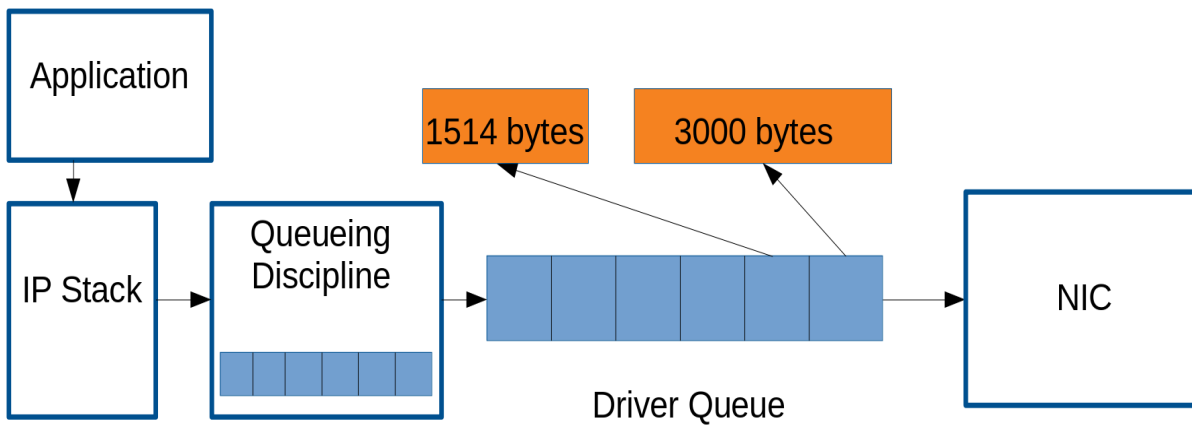


Figure 3.1: Simplified Linux network stack.

to consider as it may also alter the resources assigned to a UE abruptly, for example through slicing. Therefore, assigning the buffer size beforehand in contemporary cellular network scenarios will surely create large sojourn times. Such a problem is also present at the Linux kernel IP stack. Before the NIC, there is the so-called driver queue, where handles<sup>2</sup> to packets are accumulated and from which the NIC is fed. On the one hand, if the NIC wakes up and tries to pull data off an empty driver queue, a transmission opportunity is lost and thus, the throughput decreases. On the other hand, if too many handles to packets are present in the driver queue, a large sojourn time will occur. Due to the handles nature, packets of different sizes can be accumulated at the driver queue, and if the TSO is not disabled, packets bigger than a MTU can be accumulated, as observed in Fig. 3.1, considerably augmenting the sojourn time of all the packets.

In this scenario, Dynamic Queue Limit (DQL) [69], which is an implementation of BQL, appeared with the aim of limiting the number of bytes in the driver queue without starving it and avoiding excessive packet accumulation. After an interval of time, DQL assesses whether the hardware was starved and the queue limit was reached, in which case, the queue limit is increased. If there are still bytes to transmit in the queue, the queue limit is decreased by the number of bytes not transmitted yet [69]. After thoroughly studying this scheme, we mapped Linux's driver queue to the RLC buffer in the cellular network and proposed the DRQL.

The DRQL algorithm (cf. Algorithm 1) consists of the following variables: *limit* that represents

<sup>2</sup>A handle is a pointer to the data packet memory location along with its size in this context.

---

**Algorithm 1** DRQL pseudo-code
 

---

```

1:  $T = TTI, min\_val = INF, dequeued\_bytes = 0$ 
2:  $limit = MAX\_VAL\_LIMIT, last\_time = now$ 
3: procedure DEQUEUED(bytes)
4:    $dequeued\_bytes = dequeued\_bytes + bytes$ 
5:   update limit_reached
6:   update remaining
7:    $buffer\_starved = no\ remaining \wedge limit\_reached$ 
8:   if buffer_starved then
9:     increase buffer's limit
10:  else if remaining then
11:    if  $min\_val > remaining$  then
12:       $min\_val = remaining$ 
13:    end if
14:    if  $last\_time + T > now$  then
15:       $last\_time = now$ 
16:       $limit = max(limit - min\_val, MTU)$ 
17:       $min\_val = INF$ 
18:    end if
19:  else if  $last\_time + T > now$  then
20:     $last\_time = now$ 
21:     $min\_val = INF$ 
22:  end if
23: end procedure

```

---

the queue limit in bytes, *dequeued\_bytes* that provides the bytes that were forwarded, *last\_time* that is a timestamp since last interval,  $T$  that represents an interval during which the sojourn time of the packets is measured, and *min\_val* that saves the minimum queue occupancy during an interval. During initialization,  $T$  is set to the system TTI, *min\_val* to the maximum value supported by the variable type of the system, mimicking the identity value of the mathematical minimum function (e.g., in C language the `UINT_MAX` macro), *dequeued\_bytes* to 0, *limit* to the maximum RLC queue capacity and *last\_time* to the actual time.

The algorithm works in the following way: on the one hand, the SDAP sublayer is responsible for continuously querying the *limit* value to the RLC, as well as the current size, and deciding whether to forward more bytes or keep them. On the other hand, the MAC sublayer is responsible for calling the *DEQUEUED* procedure whenever it dequeues data from the RLC buffer, which will happen under hard real-time constraints in 5G, as MAC will call the procedure every TTI. The *DEQUEUED* procedure first checks if the limit was reached during that interval of time (line 5), and whether there are remaining packets at the queue (line 6). Next, it checks whether a queue starvation happened (line 8). If this is the case, the buffer limit is increased as starvation happened and some transmission possibilities were squandered. If the buffer limit was reached, and no more data remains at the queue (definition of *buffer\_starved*, line 7), the queue could have been provided with more data, and some bandwidth has been squandered. Therefore, the buffer limit is immediately increased. If, on the contrary, some data still remains in the buffer (line 10), and it is smaller than the *min\_val*, this value is assigned to *min\_val*. If after an interval time (line 14), there has always been remaining data, the *limit* value is reduced by the maximum between the lowest value observed during that interval, and kept at *min\_val*, and the equivalent value of a MTU (i.e.,  $\max(\text{limit} - \text{min\_val}, \text{MTU})$ ) at line 16. In this manner, the equivalent data to a MTU is maintained at the RLC buffer even for small values of *min\_val*. If the limit is not reached and there is no remaining data, as for example when the number of bytes that arrived at the RLC buffer were not enough to start accumulating, the internal timer is simply reset and the *min\_val* is set to *INF* as all the data that was forwarded into the buffer on that interval was successfully delivered to the MAC sublayer.

DRQL works in a synchronous manner due to cellular networks nature, in contrast with DQL that is asynchronous. Additionally, DRQL has to handle other cellular network particularities, such as the large channel variability or the packet segmentation/reassembly that happens at



the RLC. However, DRQL was design to map DQL's principles into the cellular network, as faithfully as possible.

### 3.3 A Queuing Theory Approach to the 5G Stack: 5G-BDP

RLC's optimal queue size can also be modeled using queuing theory as depicted in Fig. 3.2, where the SDAP and the RLC sublayer queues are involved. Based on queuing theory from [25] and [70], and using 5G's specificities, we can model  $\lambda$  as the arrival rate,  $\mu$  as the service rate of each server, and  $K$  as the number of servers in the system.  $K$  can be thought as the bits that can be transported. For example, with a 28 MCS index in a 5MHz eNodeB, approximately 2289 bytes per TTI can be transported [12], and thus, there would exist  $2289 \times 8 = 18312$   $K$  servers. Additionally, we define  $\rho = \lambda/(K\mu)$  as the utilization factor. The Little's Theorem [70] (for deterministic as well as stochastic flows) determines that the average number of customers in the queue (i.e.,  $\bar{N}$ ) results from the following equation:

$$\bar{N} = \lambda T(\rho) \quad (4.1)$$

where  $T(\rho)$  is the mean system response time (the sum of the sojourn time at the queue and server time). Since server time cannot be reduced, the minimum mean response time occurs when the sojourn time is 0, which happens when there are no customers in the queue and, therefore,  $\rho = 0$ , which implies that  $T(0) = 1/\mu$ . If we consider a deterministic arrival and a deterministic service, we can model our system as a D/D/K system. This is a simplification in a 5G system, but the deterministic approach is very useful due to the fact that according to the Law of Large Numbers [70], some conclusions may be extended to stochastic systems. As there are  $K$  servers, the total system service capacity is equal to  $K\mu$  and, therefore,  $\lambda_{\max} = K\mu$  if the sojourn time is to be minimized. The behavior of the system can be graphically understood looking at Fig. 3.3, where there exists two inaccessible regions. On the one hand, no matter how many customers ingress the system, the utilization factor  $\rho$  cannot surpass full normalized utilization (i.e., the region on the right of  $\rho(1.0)$ ). If the arriving customers rate ( $\lambda$ ) is superior to the service time ( $\mu$ ) times  $K$  (i.e., the serving rate), the mean response time ( $T(\rho)$ ), tends to infinity as customers start accumulating at the queue. On the other hand, the mean response time of the system ( $T(\rho)$ ) cannot be reduced beyond the serving time (i.e., the region below  $T(0)$ ). With these constraints, the optimal point is the  $\beta$  knee (with  $\rho = 1$ ,  $T(\rho) = T(0) = T(1.0)$ ), where the response time is the minimum while the utilization is maximum. From (4.1), for

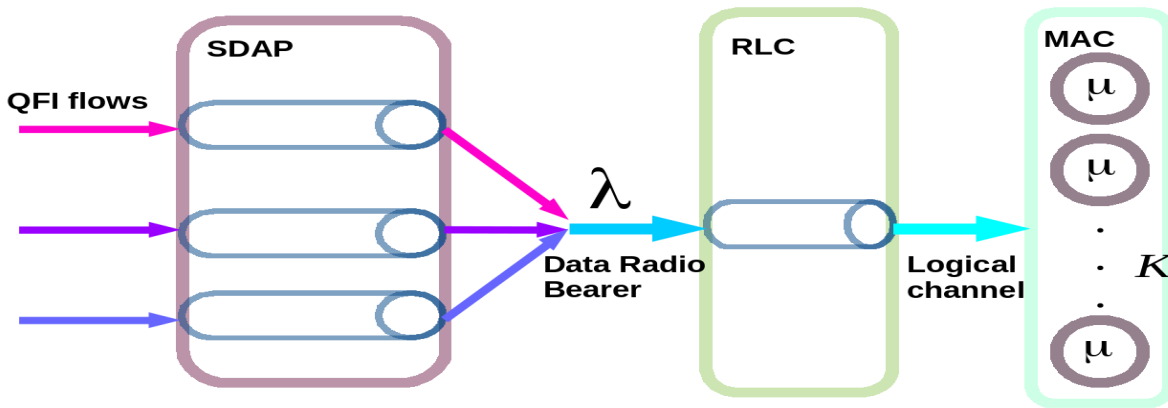


Figure 3.2: Simplified data path model for SDAP and RLC sublayers queues with multiple QFIs that converge to one DRB.

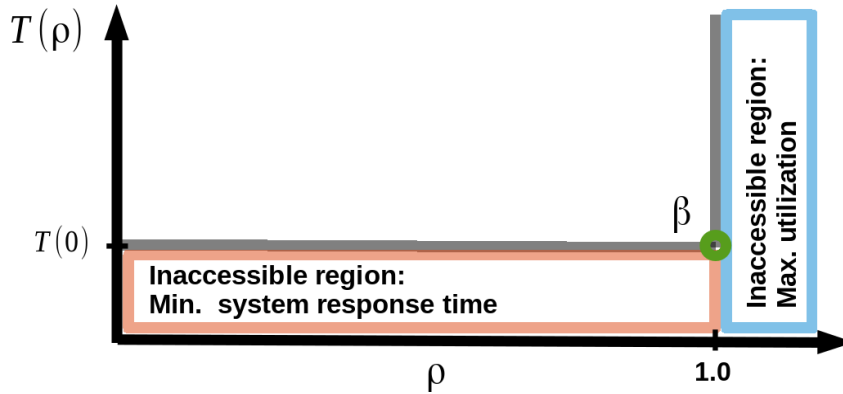


Figure 3.3: A D/D/K deterministic queuing system.

the optimal operating point with  $\lambda_{\max} = K\mu$ , we get the intuitive result of  $\bar{N}^* = K$  (i.e., no more customers than servers). Conversely, BDP (the BDP is in fact the optimal theoretical point where no queue is generated while enjoying full bandwidth as demonstrated by [52]) is defined as the Bottleneck Bandwidth (BBandwidth) times the No-Load Delay (NLDelay) (i.e., the time needed to physically traverse the path). In this case, the NLDelay is  $1/\mu$  and the BBandwidth is  $K\mu$ , and therefore, the  $BDP = K$ , which results in  $BDP = \bar{N}^*$ . This is a very valuable result that can be intuitively well understood. We want to maintain our server fully utilized, while the customers should suffer zero queue sojourn time. We are applying the principle of *keeping the pipe full, but not fuller* meticulously described by Kleinrock at [25].

In 5G systems, on the one hand the radio channel quality is delivered through the Channel

Quality Indicator (CQI) in uplink to the RAN by the UE. The CQI index is a scalar, the value of which is translated into a modulation (i.e., Quadrature phase shift keying (QPSK), 16 Quadrature Amplitude Modulation (QAM), 64 QAM or 256 QAM) and from it, to a MCS index [71]. The MCS index is then mapped to the total number of transport bytes, depending on the RBs assigned by the MAC, to assure an error rate lower than 10% [71]. With this metric, the total radio data link bandwidth can be computed. On the other hand, the radio slot length is known, and since the serving time of the packets is clearly governed by the slot duration (the physical propagation delay and the processing time of the packets can be neglected) the delay metric can be obtained. In 5G, the slot duration can vary from 1  $ms$  to 62.5  $\mu s$  at the cost of utilizing more frequency spectrum (i.e., from 15  $kHz$  to 240  $kHz$  subcarrier spacing) [72]. Since the lowest common denominator is 1  $ms$  for all the slot durations, we can calculate the optimal BDP every 1  $ms$  by multiplying the maximum slot duration (i.e., 1  $ms$ ) with the data link bandwidth. In essence, every 1  $ms$  a new D/D/K queuing system is calculated, where every server, can process the minimum number of information (i.e., one bit) per millisecond, and the number of servers  $K$  varies with the time according to the aforementioned conditions, with a serving time equal to the maximum radio slot duration (i.e., 1  $ms$ ). Modeling the arrival and the service time as deterministic is a simplification. The various retransmission mechanisms (i.e., HARQ/Negative Acknowledgment (NACK) and RLC AM), or an abrupt change in the number of RBs available per RLC buffer, can impact the sending rate. However, the conclusions obtained from deterministic model assumptions have already shown their benefits in real stochastic network deployments. For example, BBR [22] uses a deterministic queue to model the network's behavior, even though the network bandwidth and RTT are non-deterministic (e.g., a path may be shared with other flows, altering the available bandwidth and RTT). The continuously estimated BDP, eventually determines its packet's forwarding pace. In [37] different TCP congestion control algorithms in a cellular network scenarios are simulated. BBR's positive results regarding latency in comparison with other TCP congestion control algorithms in cellular networks, are heuristically confirmed. There, BBR's deterministic queuing model for cellular networks, with its certain shortcomings is validated.

Aiming to work on the optimal BDP operation point, we present a cellular BDP algorithm denoted as *5G-BDP* (cf. Algorithm 2), which works between the SDAP and the RLC sublayers, without requiring any information from the MAC sublayer.

During the initialization, the variable *offset* is set to a value in the range of  $[0,1.0)$ . This value is aggregated to the normalized TTI value to reduce the starvation possibility (line 15). The normalized TTI represents the normalized elapsed time since last TTI (i.e., it will be in the range of  $[0,1]$ ). The variables *bytes\_to\_send* and *bytes\_tx* are set to 0 and the *last\_time* to *now*.

---

**Algorithm 2** 5G-BDP Product pseudo-code
 

---

```

1: offset = NORMALIZED_OFFSET
2: bytes_tx = 0, bytes_to_send = 0, last_time = now
3: procedure SET_VALUES (remaining, channel_capacity)
4:   if channel_capacity > remaining then
5:     bytes_to_send = channel_capacity – remaining
6:   else
7:     reset bytes_to_send
8:   end if
9:   reset bytes_tx
10:  last_time = now
11: end procedure
12: function GET_OPTIMAL_VALUE
13:  update elapsed_time
14:  update normal_tti
15:  normal_tti = normal_tti + offset
16:  update soll_tx
17:  if soll_tx > bytes_tx then
18:    return soll_tx – bytes_tx
19:  else
20:    return 0
21:  end if
22: end function

```

---

MAC sublayer calls the *SET\_VALUES* procedure every TTI to calculate the optimal number of bytes to forward (i.e., *bytes\_to\_send*), to reset the number of packets transmitted in the last interval (i.e., *bytes\_tx*), and to save the actual time. For this, the CQI index received from the UE together with the last number of used RBs are converted into bytes, which is passed to the function through the *channel\_capacity* variable, together with the remaining number of bytes at the

RLC buffer (i.e., *remaining*). The number of remaining bytes in the queue informs the algorithm how many bytes are accumulated at the RLC buffer. To forward the packets at a correct pace from the SDAP buffer to the RLC buffer, the SDAP sublayer calls the *GET\_OPTIMAL\_VALUE* function periodically with a period smaller than a TTI. It calculates the normalized value of the time elapsed since the last time that new values arrived (i.e., *elapsed\_time*) at the *SET\_VALUES* procedure, and determines how many bytes could have been forwarded from the SDAP towards the RLC within this duration (i.e., *soll\_tx*). If the amount of already transmitted bytes exceeds this latter value, no packets are forwarded to the RLC buffer. Otherwise, the difference between the number of bytes that could have been transmitted and the actual amount of transmitted bytes is returned (line 18). With this information, the SDAP determines how many packets can be dequeued. There might exist a difference between the estimated BDP and the real BDP due to diverse factors (e.g., HARQ/NACK retransmissions). 5G-BDP minimizes the impact of this difference, by using the remaining number of bytes at the RLC buffer to calculate the *bytes\_to\_send* every TTI. In this way, the queue formed when the BDP is overestimated (i.e., the real BDP is smaller than the calculated BDP and more data than the data pulled by the MAC has been forwarded), is reduced in the next TTIs, as no more data will be forwarded until the buffer is depleted below the *channel\_capacity* as described at Algorithm 2 in procedure *SET\_VALUES*.

5G-BDP differs in various ways from DRQL. DRQL defines a limit, which controls the amount of data to pass to the RLC. However, unlike 5G-BDP, this limit is not updated based on the BDP. Instead, it uses a heuristic to update that limit value based on the buffer state. 5G-BDP, on the contrary, uses the channel capacity value and the RLC buffer state to control the SDAP-RLC data transfers. In addition, 5G-BDP uniformly distributes the submission of packets to the RLC within a TTI, which defines a pacing mechanism at SDAP allowing prioritization of low-latency traffic arriving during that TTI. To this end, 5G-BDP forwards packets according to the number of bytes that could have been used for the elapsed time since last TTI (e.g., if 0.5 ms elapsed since last TTI and the BDP is 2200 bytes/ms, 1100 bytes can be forwarded).

### 3.4 Enhanced 5G-BDP: e5G-BDP

5G-BDP correctly calculates the BDP, which is the optimal pacing value at which the packets should be forwarded to work on the optimal point [25] [73], not starving the queue and avoid-

ing bloating it. 5G-BDP relies in last TTI's bandwidth to forward a packet. This translates to perform poorly when the RBs distribution largely varies between TTIs. In essence, it performs better if the MAC scheduler distributes the RBs quasi-uniformly, such as  $\{6,7,6,6,5\}$  instead of  $\{0,12,0,0,18\}$  during 5 TTIs. However, such a decision relies on the MAC scheduler algorithm from which 5G-BDP is decoupled, and therefore, lacks this information. Additionally, the MAC scheduler algorithm may assign RBs according to the actual buffer occupancy. In such scenarios, where the RBs are not uniformly distributed, a non-virtuous cycle may occur. 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 will not assign more RBs to the RLC buffer due to its low occupancy. To avoid the situation where the SDAP sublayer does not forward more packets to prevent the bufferbloat, while the MAC scheduler does not fetch more bytes from the RLC buffer due to its low occupancy, the e5G-BDP solution 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 may also be forwarded according to the BDP, yet ignoring the amount of accumulated bytes at the RLC buffer. As 5G-BDP, 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  $\mu$ s at the expense of using more frequency spectrum [72]. Lastly, 5G-BDP does not consider the size of the current packet to transmit. Once a large packet has reached the RLC buffer, completely transmitting it can last several milliseconds, specially in scenarios with low throughput. Therefore, e5G-BDP delays forwarding large packets within the TTI, and thus, improves the sojourn time suffered by low-latency packets as pictorially represented in Fig. 3.4. Note that e5G-BDP is a pacer. Thus selecting the next packet is the scheduler's responsibility, and thus, e5G-BDP does not decide whether the next packet should be large or small. It just answers the question of whether the next packet should be forwarded trying to avoid the bufferbloat, considering the elapsed time since last TTI, the estimated bandwidth, the packet size and the RLC buffer status.

The pseudo-code for e5G-BDP is presented through the Algorithms 3, 4 and 5. Algorithm 3 represents the main e5G-BDP function, which is called by the SDAP scheduler per active RLC

buffer within a TTI periodically, in order to determine whether a packet can be forwarded to the subsequent lower sublayer (i.e., RLC) or it should be retained. It firstly checks whether the *update\_flag* variable is set (line 2), and if so, it recalculates the bandwidth calling the function *update\_last\_bandwidth* given in Algorithm 5. The *update\_flag* is set every 1 ms once the packets from the RLC are forwarded. At line 6, it is checked if the sum of the already SDAP-RLC submitted bytes (i.e., *srs\_bytes*) and the last accumulated bytes 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 was reached, and therefore, no more packets are forwarded. If the aforementioned sum is smaller than the capacity, the *paced\_bytes* (line 10) are calculated. This feature, enables the pacing capability, as the theoretical transmitted bytes depend on the elapsed time since the last 1 ms TTI (i.e., 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 paced bytes without any multiplicative or additive factor). At line 11, the *paced\_bytes* value 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 transmitted 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 (i.e., *last\_acc\_bytes*), contrarily to what happens at line 15 mostly in the latter moments of the TTI (i.e., in the (500, 1000)  $\mu$ s range rather than the (0, 500)  $\mu$ s range since last transmission, assuming a 1 ms TTI), as the value of *paced\_bytes* increases with the elapsed time and, therefore, dispatches packets more actively as compared with 5G-BDP. It also favors forwarding small packets. At line 14, the *extra\_bytes* variable that depends on the next packet size, the current transmitted bytes and the last size of the RLC buffer, is created. It incentives transmitting 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 (i.e., *extra\_bytes*) also discourages transmitting large packets as it depends on the packet's size, which are one of the causes of the bufferbloat. Note that e5G-BDP is a pacer, and thus, only responsible to forward packets at an appropriate pace to avoid the bufferbloat. A scheduler is responsible for deciding which is the candidate packet among the available possibilities and informing the pacer. Thus, the scheduler is responsible for deciding which packet should be scheduled next and which fairness is maintained between the DRBs. e5G-BDP avoids bloating the buffers of diverse QFI that share the same DRB, and hence, once they are submitted to the RLC, they cannot be further segregated. Lastly, at line 15, the already sent 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 forward or keep the packet at the SDAP sublayer.

---

**Algorithm 3** e5G-BDP: *limit\_reached*


---

```

1: function LIMIT_REACHED(data_size)
2:   if update_flag == True then
3:     update_bw_est() /* Update the BW */
4:     update_flag = False
5:   end if
6:   if srs_bytes + last_acc_bytes > max_bytes then
7:     /*If transmitted bytes from SDAP to RLC + last measured occupancy above the max.
       number of bytes that can be transported in one TTI, limit reached*/
8:     return True
9:   end if
10:  paced_bytes = calculate_paced_bytes()
11:  if paced_bytes*reduce_const > data_size + srs_bytes then
12:    return False
13:  end if
14:  extra_bytes = tx_bytes != 0  $\wedge$  last_acc_bytes != 0 ? data_size/3 : data_size/5
15:  if srs_bytes + last_acc_bytes + extra_bytes > paced_bytes then
16:    /*If transmitted bytes from SDAP to RLC + last measured occ. + extra bytes above
       calculated paced bytes, limit reached*/
17:    return True
18:  end if
19:  return False
20: end function

```

---

Algorithm 4 and Algorithm 5 are helper functions where, the last bandwidth according to the RLC buffer occupancy after the 1 ms TTI is estimated, and the theoretical bytes that should had been transmitted are calculated (i.e., *paced\_bytes*), respectively. At Algorithm 4, the bandwidth is estimated according to the number of bytes that were pulled by the MAC from the RLC buffer through an EWMA calculation. Besides, the accumulated bytes (i.e., *last\_acc\_bytes*) and the last 1 ms TTI variables (i.e., *last\_acc\_bytes* and *srs\_bytes*) are updated.



---

**Algorithm 4** e5G-BDP: update\_bw\_est

---

```

1: procedure UPDATE_BW_EST
2:   acc_bytes = get_bytes_rlc_queue()
3:   rms_bytes = srs_bytes - (acc_bytes - last_acc_bytes) /*RLC to MAC submitted bytes*/
4:   bandwidth = EMWA(rms_bytes/TTI) /*Exponential Moving Weighted Average*/
5:   srs_bytes = 0
6:   last_acc_bytes = acc_bytes
7:   last_tti = now
8: end procedure

```

---



---

**Algorithm 5** e5G-BDP calculate\_paced\_bytes

---

```

1: function CALCULATE_PACED_BYTES(now)
2:   elapsed = now - last_tti
3:   incr = elapsed/TTI < 0.5 ? 1.2 : 1.33
4:   paced_bytes = 0
5:   if bandwidth ≠ 0 then
6:     paced_bytes = incr * elapsed * bandwidth + MTU/7
7:   else if last_acc_bytes == 0 ∧ elapsed > 0.5 then
8:     paced_bytes = MTU/4
9:   end if
10:  return paced_bytes
11: end function

```

---

At Algorithm 5 the *paced\_bytes* is calculated based on the elapsed time, the bandwidth and the last accumulated bytes (cf. Algorithm 5 in line 3). A multiplicative factor variable at line 3 (i.e., *incr*) that depends on the elapsed time is used to forward packets more actively in the last instants of the TTI, since the value increases during the second half of the TTI. An additive factor of  $MTU/7$  is used to avoid a possible starvation when the bandwidth is small yet not zero. 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 (i.e., MTU of 1500 bytes), and User Datagram Protocol (UDP) packets, that are normally scarce, small in size and do not contribute as severely to the bufferbloat as the bigger ones. In this manner, small packets are favored over large packets as they prevent the bufferbloat, yet they transmit information which has the same importance as large packets as they rely on the same DRB. Note that e5G-BDP is a pacer, and it is the scheduler's responsibility to select the next packet to forward. Thus even if the bandwidth value is small, at least a  $MTU/7$  in bytes is returned.

In the case that the bandwidth is 0, no packets are accumulated at the RLC buffer and the elapsed time is above 0.5 (i.e., line 7), the MTU value is divided by 4 (line 8) with the intention of forwarding the packet to the RLC buffer. This assures that a packet will be submitted to the RLC buffer as the theoretical calculated value (i.e.,  $MTU/4$ ) will be larger than the  $data\_size/5$  for all the packet sizes, in cases where the *last\_acc\_bytes* is 0 and the transmitted bytes (i.e., *srs\_bytes*) is also 0, as seen in Algorithm 3 at line 13.

e5G-BDP pacing mechanism is illustrated in Fig. 3.4. 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) [74] 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. As observed in Fig. 3.4, 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  $1.2 \times 0.25 TTI \times 1000 bytes/TTI + 1500/7 bytes = 514$  bytes, while

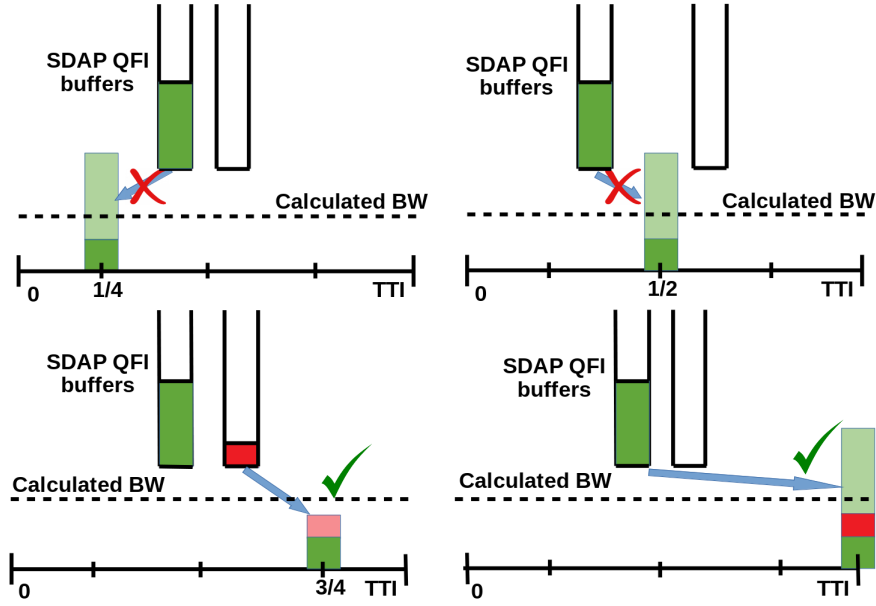


Figure 3.4: Possible e5G-BDP pacing mechanism outcome.

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.,  $paced\_bytes = 1.2 \times 0.5 TTI \times 1000 \text{ bytes}/TTI + 1500/7 \text{ bytes} = 814$  bytes), since forwarding a large packet during the first moments of the TTI, would block the possibility of other packets that may arrive during the TTI to be transmitted in the next TTI. Such effect of sharing queues among flows with diverse QFIs appears due to the funnel nature of 5G, as explained in Section 2. 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  $t = 3/4$ . The scheduler, now decides that the next packet to transmit is the red small packet. Since *paced\_bytes* increases with the elapsed time (i.e.,  $1.33 \times 0.75 TTI \times 1000 \text{ bytes}/TTI + 1500/7 \text{ bytes} = 1211.5$  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 informs the scheduler that the large packet that it refused to submit before can be now forwarded, 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.,  $1.33 \times 1.0 TTI \times 1000 bytes/TTI + 1500/7bytes = 1544$  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. 3.4, if we suppose that the next bandwidth will be equal to the calculated one (i.e., 1000 bytes/TTI), 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.

e5G-BDP differs in the same ways as 5G-BDP from DRQL [73] presented in Section 3.2. DRQL lacks a pacing mechanism and, therefore, packets are forwarded in a bulky manner, reducing the opportunity for prioritization of low latency flows that arrive during a TTI. Additionally, DRQL does not fit well within a fluctuating radio link channel (e.g., dynamic MCS scenario). DRQL relies on keeping an equivalent amount of bytes as the Ethernet MTU size at the RLC buffer, which may also present problems in a dynamic scenario. In summary, DRQL relies on the amount of bytes at the RLC buffer, while e5G-BDP uses the elapsed time since last TTI, and the amount of bytes transmitted to change its rate, allowing it to adapt to the dynamic conditions and avoiding bloating the RLC buffer.

### 3.5 Pacing the Traffic between the UPF and the SDAP: UPS

If a containment mechanism is implemented at the RLC buffer, the bufferbloat will be transferred to the next queuing level as demonstrated in [68]. If, on the contrary, no containment mechanism is implemented at the RLC sublayer, any attempt to tackle the bufferbloat problem at the SDAP will be superfluous. Therefore, any queue management algorithm proposed at the SDAP sublayer must be combined with a containment mechanism at the RLC in order to be effective. However, the problem description differs slightly from the (e)5G-BDP approach. In the first place, the communication time between the UPF and the SDAP may not be negligible, challenging the deployment. Additionally, the radio link dynamicity certainly demands a low-latency communication between both entities if an accurate value of the real throughput is to be considered, and thus, avoid the bufferbloat. However, the principle of *keep the pipe full*,

*but not fuller* also holds. For scenarios where a quick communication in comparison with the TTI between the UPF and the RAN can be established, the *UPF-SDAP Pacer (USP)* algorithm has been developed. USP forwards the packets from the UPF to the SDAP at a rate that avoids building large queues at the SDAP sublayer, while maintaining its buffer always occupied.

---

**Algorithm 6** USP: dequeued
 

---

```

1:  $T = TTI, num\_ticks = 0, dequeued\_bytes = 0$ 
2:  $dequeued\_bytes\_last = 0$ 
3:  $saturation\_detected = False$ 
4:  $optimal\_occ = MAX\_OCC$ 
5: procedure DEQUEUED(bytes)
6:   if  $saturation\_detected$  then
7:      $saturation\_detected = False$ 
8:     calculate  $optimal\_occ$ 
9:      $last\_time = now$ 
10:     $dequeued\_bytes\_last = dequeued\_bytes$ 
11:    reset  $num\_ticks$ 
12:  else if  $now > last\_time + T$  then
13:     $last\_time = now$ 
14:     $dequeued\_bytes\_last = dequeued\_bytes$ 
15:    reset  $num\_ticks$ 
16:  end if
17:   $dequeued\_bytes = dequeued\_bytes + bytes$ 
18: end procedure

```

---

Algorithm 6 and 7 illustrate our proposal.  $T$  represents a time interval, and  $num\_ticks$  is the number of times that the UPF queried the SDAP sublayer during the last  $T$  interval. The number of dequeued bytes so far is saved at  $dequeued\_bytes$ , while the amount of dequeued bytes at the last  $T$  interval are saved in the  $dequeued\_bytes\_last$  variable. The  $saturation\_detected$  represents a flag to inform if the RLC buffer stopped accepting data, and the  $optimal\_occ$  represents the number of estimated bytes that the SDAP buffer should have in every moment. In essence, it should have enough bytes to rapidly transmit them to the RLC, while avoiding the bufferbloat. The UPF network function asks to the USP about the SDAP's actual size, as well as the optimal SDAP occupancy (cf. Algorithm 7). In case that the current occupancy is below the

**Algorithm 7** USP: *get\_opt\_occupancy*


---

```

1: function GET_OPT_OCCUPANCY
2:   increment num_ticks
3:   if now > last_time +  $T/2$  then
4:     return clamp(incr * optimal_occ)
5:   end if
6:   return optimal_occ
7: end function

```

---

optimal occupancy, enough packets to reach the optimal occupancy are sent. Every time that the UPF executes a query to get the optimal occupancy (cf. Algorithm 7), the *num\_ticks* value at USP is increased (Algorithm 7 line 2). It has to be noted that this value may suffer deviations in soft real-time environments, since the deadlines are non-deterministic in such environments. On the other hand, the SDAP sublayer will set the *saturation\_detected* flag to *True* once the RLC buffer reaches a congested state. Once the RLC buffer accepts more data again, SDAP will dequeue data from its queue and the *DEQUEUED* procedure (cf. Algorithm 6 line 5) of the USP algorithm will be called.

The *DEQUEUED* procedure will compare whether a saturation state was reached in the previous TTI (cf. Algorithm 6 line 6). If this is the case, the flag will be reset and a new optimal occupancy value for the SDAP queue will be calculated (cf. Algorithm 6 line 8) according to the bytes accepted by the RLC buffer in the previous interval and the number of times that UPF asked to the SDAP for the *opt\_occupancy* of the queue (Algorithm 7). The optimal occupancy of the queue is doubled after half of the TTI is consumed (i.e., 0.5 *ms* for LTE systems where the TTI lasts 1 *ms*) (cf. Algorithm 6 line 5), adopting a more aggressive pacing strategy than 5G-BDP, and it will just be doubled once every *T*. This mechanism is an heuristic that assures feeding the SDAP sublayer with enough bytes, even in the case where, due to the soft real-time environment, the query time from the UPF may suffer deviations. It also assures that in case that no saturation was detected in the *DEQUEUED* procedure, a saturation state will be found in the near future if enough packets are forwarded. The *optimal\_occ* value is clamped not to grow indefinitely in case where no saturation is reached, or to fall to zero in case that no packets were dequeued.

This algorithm differs from (e)5G-BDP in various ways, even though the idea of maintaining the

queue *full but not fuller* [25] remains. In the first place, it is of vital importance not to starve the SDAP queues, since that would also starve the RLC queues and ultimately reduce the throughput. Therefore, in a soft real-time communication system as the one between the UPF and the SDAP (i.e., UPF may not always have the same amount of transmission opportunities), USP carries a more aggressive packet forwarding pacer than (e)5G-BDP to avoid starving the queue. Hence, the SDAP queues are always slightly bloated. Secondly, *no a priori* information about the bandwidth is shared. USP is not provided with more information other than the actual status of the SDAP queues, and the number of times that UPF queried the SDAP sublayer during the last interval time. With this data, it estimates the optimal queue occupancy at the SDAP sublayer.

## 3.6 Evaluation Frameworks and Experimental Results

To validate the proposed algorithms in this chapter, a 5G QoS hierarchical queuing emulator was developed from scratch<sup>3</sup>. After confirming its validity, a step forward was taken and the implementation was migrated into a real cellular network testbed, to confirm the correct operation of the (e)5G-BDP solution. In the following, a thorough description of the emulator, as well as the testbed, alongside their corresponding results are presented.

### 3.6.1 5G QoS Hierarchical Queuing Emulator

A 5G QoS hierarchical queuing emulator following the Fig. 2.4 (i.e., UPF, SDAP and MAC entity/sublayers) was implemented in C from scratch. In the emulator, IP packets are redirected from the kernel space to the user space for emulating 5G's QoS hierarchical queues. We use the NFQUEUE traffic control *netfilter* queue binding. All the input packets are forwarded to the user space through an *iptables* rule, where the QoS solutions presented in Table 3.1 (i.e., Vanilla, (e)5G-BDP, DRQL, DynRLC, CoDel, BBR and USP) are applied, including the packet forward/-drop decisions, since AQM algorithms may decide to drop a packet to inform the congestion control TCP algorithm that excessive packet accumulation is taking place. For BBR, we use the exact same scenario as Vanilla and changed the default TCP congestion control algorithm Cubic to BBR, while we use Cubic for the rest of the scenarios. Selecting the TCP congestion control algorithm to transport the data may for example appear in a scenario where data is located in a close MEC server. Every entity/sublayer (i.e., UPF, SDAP and MAC) runs in a separate thread

---

<sup>3</sup>The code can be freely downloaded from <https://github.com/mirazabal/Dynamic-buffer-TMC>.

that executes in an infinite loop.

When the flows arrive from the DN to the 5G network emulation Personal Computer (PC), they could be hashed using the 5-tuple (source/destination IP address, source/destination port and type of traffic) as the key for the Jenkins hash function and classified as an IP tuple flow. A SFQ [47], minimizes a possible collision between different flows while providing an efficient solution. The UPF network entity gets a transmission opportunity every  $1\text{ ms}$ , where a maximum of 10 packets can be forwarded to the SDAP. As mentioned in Section 2, the SDAP sublayer maps UPF marked packets into RLC buffers based on their QFI. Since the egressing scheduler is not specified by 3GPP, we implement a priority scheduler where low-latency packets are firstly dequeued, with a capability of egressing 10 packets among active SDAP queues and its priority every millisecond. The packets are finally forwarded into the RLC buffer, where they wait until the MAC sublayer pulls every  $10\text{ ms}$  the amount of bytes requested by the PHY layer. We implemented the MAC scheduler as a Round Robin scheduler, where the number of bytes that can be egressed is determined by the radio channel conditions according to [26]. Since the bottleneck resides at the MAC rather than at SDAP or UPF, it was decided to create a ten to one relationship between the forwarding capabilities of the UPF and SDAP in comparison with the MAC sublayer. The fact that the emulation was run on a soft real-time system prevents the adoption of forwarding packets every  $1\text{ ms}$  at MAC sublayer, which is the TTI for LTE cellular systems, and  $0.1\text{ ms}$  at SDAP and UPF. In any case, the results generated follow the same trend if the MAC sublayer forwards packets every  $1\text{ ms}$  or  $10\text{ ms}$ , as demonstrated in [75]. With these parameter values, we successfully emulate the realistic scenarios where the packets tend to accumulate at the lower entities since the radio data link is the principal bottleneck of cellular systems.

In this emulator, we define two scenarios (cf. Fig. 3.5). In the first scenario, two IP flows belonging to two QoS classes (i.e., two different QFIs) are mapped into 2 different SDAP queues that are lastly mapped into one RLC buffer. This is the case for services with different QFIs that share a RLC buffer. As mentioned in Chapter 2, services with different QFIs will inevitably share RLC buffers, and it is of capital importance to avoid the bufferbloat problem if different constraints of diverse services want to be fulfilled. In the second scenario, we mapped two different flows into one SDAP queue that is mapped into one RLC queue. Due to the large number of flows that are nowadays generated and its dynamic nature, segregating all flows to distinct QFIs is unrealistic, due to the limited QFI range. Therefore, some non-critical low-latency traffic will end up in a



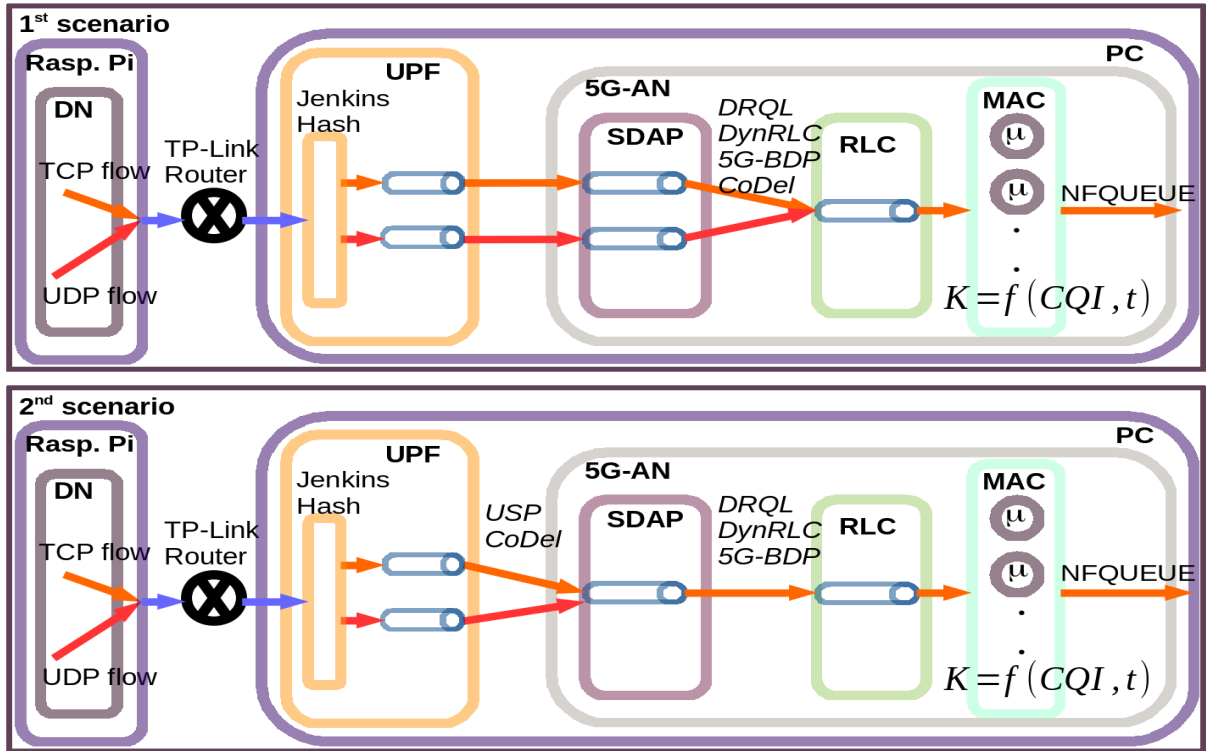


Figure 3.5: Evaluation framework with first and second scenarios.

scenario, where the created delays of shared queues may downgrade the user experience due to the bufferbloat. If the communication between adjacent sublayers/entities in the data path (e.g., UPF-SDAP) can be performed within a TTI, a pacing algorithm that gradually forwards the packets between these sublayers/entities can be implemented. This increases the chances of delivering a newly arrived delay constrained packet through the different queues without suffering large sojourn times. We implement the USP algorithm between the UPF and the SDAP queues and combine it with the three methods aforementioned: DynRLC, DRQL and 5G-BDP at the RLC queues. Furthermore, in case where such a communication is not possible (e.g., splitted 5G scenarios or any standard setup where UPF and RAN are separated), we place the well known AQM algorithm CoDel at the SDAP buffers and combine it with DynRLC, DRQL and 5G-BDP at the RLC buffer, to indicate to the transport layer's congestion control algorithm of the sender that excessive buffer accumulation is happening. A CoDel-CoDel combination is not presented as CoDel is not a backlog mechanism and, thus, all the packets would flow to the RLC, leading to the same results as in the first scenario, where the packets are accumulated at the RLC buffer. The competing traffic flows (i.e., one with low-latency requirements and one with bursty traffic) emulate a scenario where different flows try to access the scarce resources

in a saturated scenario. The low-latency traffic is modeled by UDP datagrams that emulate the traffic of online gaming applications [76]. Based on the traffic characteristics of such applications derived from real traces [26], we generated packets with a range of average interval times following a normal distribution and a standard deviation of 5 *ms*. The average interval times evaluated are [20, 40, 60, 70] *ms*. The competing bulky flow is modeled using a TCP flow generated by the *iperf3* software with a MTU of 1500 bytes. The choice of TCP as the competing flow resides in the fact that most Internet traffic is forwarded through Hypertext Transfer Protocol 2 (HTTP/2), which relies on TCP [77] as its transport layer protocol. To achieve the steady state, avoid the TCP slow start, and assure that congestion happened, the bulky traffic is generated 5 seconds before the UDP datagrams. In this manner, we emulate the worst bufferbloat scenario. Fortunately, the scenarios where TCP has not reached the steady state can be conjectured to present less latency delay.

Table 3.1, resumes the algorithms tested in the 5G emulator with its major characteristics. Vanilla represents the algorithm where no action is taken (i.e., an implementation in accordance with 3GPP). 5G-BDP, DRQL and USP are our proposed algorithms, while DynRLC has been presented in [23], CoDel in [41] and BBR in [22]. To optimize the number of packets at the RLC queue, and thus reduce the delay, the cross-layer communication plays a major role as the information can rapidly flow between close entities. This characteristic makes 5G-BDP, DRQL and DynRLC significantly faster in comparison with BBR. As previously mentioned, a pacing algorithm augments the possibilities of delivering a high priority packet within the actual TTI, giving an advantage to 5G-BDP over their competitors. CoDel is announced as a knob-less solution. As explained in Chapter 2, CoDel is governed by two variables that describe the maximum desired sojourn time (i.e., *target time*) and the interval time (i.e., *interval time*). In the CoDel Request For Comments (RFC) [41], it is recommended to set the *target time* value to 5 *ms*, although no theoretical background for such a number is given but rather a heuristic. Even though such value has been proved to work correctly on the wired domain, and in some open source IEEE 802.11 implementations has been adopted as the default AQM algorithm [42], it does not address 5G specificities. Additionally, in the presented emulation scenario, and since the MAC egresses packets every 10 *ms* in discrete time, the *target time* of 5 *ms* recommended for CoDel [41] would hardly be reached when CoDel is implemented directly at the RLC buffer even when the queue is empty. Therefore, we increased the *target time* to 15 *ms*, which is 5 *ms* above our emulated 10 *ms* TTI, and the *interval time* to 300 *ms*, fulfilling the recommendation

Table 3.1: Algorithms tested to validate the bufferbloat avoidance solutions at the emulator.

	Vanilla	5G-BDP	DRQL	DynRLC	CoDel	BBR	USP
X-layer comm.	No	Yes	Yes	Yes	No	No	Yes
Pacing algorithm	No	Yes	No	No	No	No	Yes
Knob-less	Yes	Yes	Yes	No	No	Yes	Yes
Discard packets	No	No	No	No	Yes (or ECN)	No	No

of maintaining the *target time* to 5% of the *interval time*. On the other hand, we have maintained its default values when the CoDel algorithm is used at the SDAP queues. CoDel has been implemented from scratch following the pseudo-code provided by [41]. The DynRLC algorithm, similarly has an interval value, as well as a desired value as explained in Chapter 2. For simulating the DynRLC [23] algorithm, we set the interval value to 100 *ms* and the desired sojourn time to 11 *ms*. A lower desired value would imply that the packets that arrive at the RLC buffer could stay less than one TTI as the MAC scheduler egresses the packets every 10 *ms*. DynRLC adjusts its queue size through steps every interval time. The step size is determined by a  $\alpha$  value that we set to 0.1. The *MAX\_VAL\_LIMIT* was set to 1024 packets for the DRQL (cf. Algorithm 1) and *MAX\_OCC* was set to 1024 packets for the USP algorithm (cf. Algorithm 6) to discard an overflowing buffer effect from the study. The *NORMALIZED\_OFFSET* was set to 0.33 in the 5G-BDP (cf. Algorithm 2) to avoid a starving fatal condition at RLC in a soft real-time environment as our emulation scenario. These values have been heuristically proved to be efficient. We also present USP, which is an algorithm to be placed between the UPF and the SDAP entities. USP was designed with the same principle as (e)5G-BDP (*keep the pipe full, but not fuller*), and, therefore, it shares its main characteristics with (e)5G-BDP. Lastly, CoDel is the only algorithm that discards packets to inform the congestion control algorithm that excessive packet accumulation is happening, forcing it to resend packets from the transport layer.

In summary, for the first scenario, the Vanilla scenario, the Vanilla scenario with TCP BBR and referred to as BBR, and the DRQL, DynRLC and 5G-BDP that rely in information exchange between the SDAP and the RLC are tested. The RLC FIFO queue is replaced by a CoDel queue for the CoDel scenario. Regarding the second scenario, the Vanilla and the BBR scenarios are

used, while all the permutations of combining USP and CoDel with DRQL, DynRLC and 5G-BDP are tested. All the combinations use TCP Cubic except for BBR's scenario.

A Raspberry Pi is used to emulate the DN while the 5G QoS emulator runs in a PC as seen in Fig. 3.5. The 5G QoS emulation PC contains an Intel(R) Core(TM) i7-7500U CPU @2.70GHz running Ubuntu 16.04, while the DN is implemented on a Raspberry Pi Model 3 B+ with a Broadcom BCM2837B0, Cortex-A53 64-bit SoC @1.4 GHz. The DN generates both traffic flows, while the QoS multiqueuing emulation software runs on the 5G QoS emulation PC. A TP-LINK TL-WR841N router connects the DN with the 5G QoS emulating PC through an Ethernet connection.

For realistic evaluations, we use LTE traces provided by [26], where the reported CQIs are converted to data link rates (i.e., they represent the  $K$  value from Fig. 3.2), that is a function of time and CQI as explained in Section 3.3. At [26], statistics from the base stations are reported in a granularity of 1 second for 15 minutes and five different cases (i.e., bus, car, train, pedestrian and static). From these 5 cases, we select the pedestrian and the train cases as they represent two completely different circumstances to which the network will be exposed. Since the conversion from the CQI index to the MCS index is manufacturer specific, we employ the well tested values from OAI<sup>4</sup>. However, due to NFQUEUE's limitation, the data link throughput is calculated in packets rather than bytes. For the evaluations, the first 200 seconds of the traces are used, which correspond to 200 CQI updates and 20000 MAC scheduling opportunities per run (MAC scheduler egresses packets every 10 *ms* in our emulator). A large continuous simulation was chosen (i.e., 200 seconds) rather than many short ones to consider large CQI variations, as well as to observe long term effects of some algorithms (e.g., BBR recalculates each state every 10 seconds).

Lastly, scalability is tested with two low-latency flows that compete for accessing the resources with a greedy TCP flow with the pedestrian and the train datasets. The constraint of using two low-latency flows emerges from the fact that NFQUEUE forwards packets, rather than bytes. Therefore, if more flows are generated, the bandwidth that is estimated in packets, would get distort and unrealistic outcomes would emerge. This problem does not happen in the real testbed, were up to 8 low-latency flows were successfully tested.

---

<sup>4</sup>Conversion function `cqi_to_mcs` can be found at `openair1/PHY/phy_vars.h` [17]

### 3.6.2 5G QoS's Emulator Experimental Results

This section is divided into two parts. In the first part, we present the results of the first scenario, where two SDAP buffers are mapped into one RLC buffer. This scenario emerges when both flows share a DRB at the RLC. We name this first scenario *Individual SDAP buffer, shared RLC buffer*. In the second part, we present the results of the second scenario, and we name it *Shared SDAP buffer, shared RLC buffer*, where both flows share the SDAP as well as the RLC queue. This scenario emerges when no dedicated bearer was generated by the CN and the packets are not segregated according to their QFI in different queues. Both scenarios can be observed in Fig. 3.5.

#### First Scenario: Individual SDAP Buffer, Shared RLC Buffer

To quantitatively compare different QoS solutions, we assess the average delay of packets from a low-latency flow from the moment that enters the UPF entity until it is forwarded by the MAC sublayer, and the average queue size of RLC. The results for the pedestrian dataset are presented in Fig. 3.6 for the next algorithms: Vanilla (i.e., the basic scenario described by 3GPP with a priority scheduler at the SDAP sublayer), CoDel, BBR, DynRLC, DRQL and 5G-BDP. As it can be observed in Fig. 3.6, a direct correlation between the queue size and the delay that a low-latency packet suffers exists. The algorithms that accumulate fewer packets at the RLC queue, suffer lower delays as the sojourn time is reduced, while the algorithms that accumulate more packets suffer larger sojourn delays. This is a natural outcome from queuing theory that Fig. 3.6 confirms. The avid reader may have also noticed that in Fig. 3.6, BBR and CoDel present similar queue sizes while BBR suffers from larger delay. This fact can be explained from the following Fig. 3.7, where BBR clearly shows a larger deviation when compared to CoDel. Even though they have similar average queue sizes, BBR during the first seconds, where the radio link channel capacity is scarce, presents larger queues, while CoDel presents larger queues during the last seconds. Having larger queues while the radio link channel capacity is scarce, results in needing more TTIs in comparison when the channel capacity augments during the last seconds.

In Fig. 3.7, the radio link channel capacity profile from the pedestrian dataset can be observed in red color (i.e., the previously represented  $K$  number of servers in Fig. 3.5). This profile represents the maximum number of packets that the PHY layer can pull. In green color, the size of the RLC buffer during the emulated 200 seconds can be observed with all the algorithms

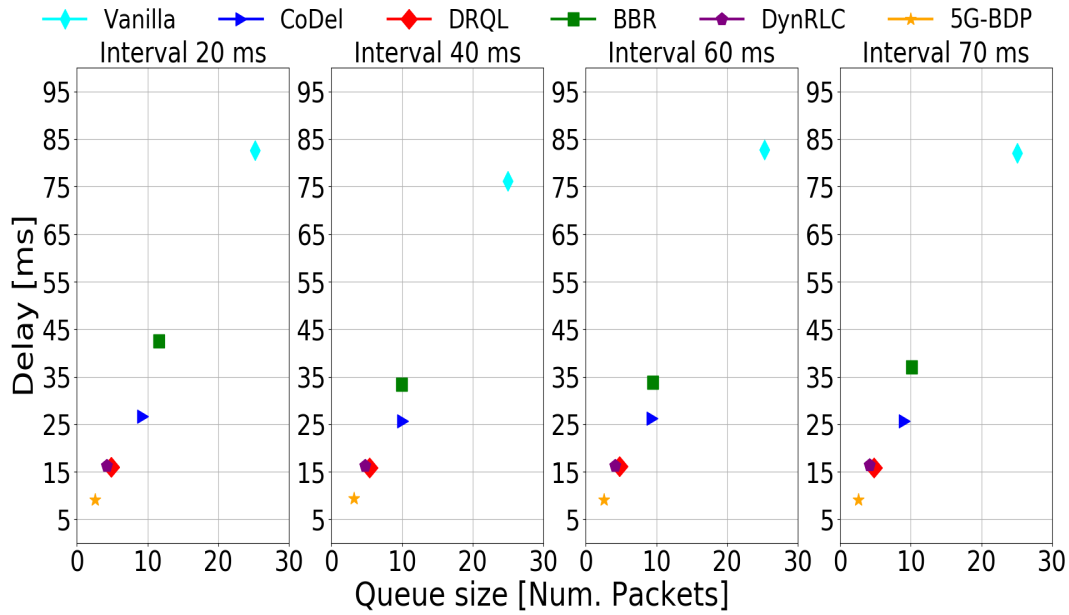


Figure 3.6: Average queue size and average delay for pedestrian dataset.

tested. If no countermeasures are applied (i.e., Vanilla case), excessive packet accumulation happens due to TCP's greedy congestion control nature, but no resources are squandered. The RLC buffer always possesses enough packets to forward to the MAC sublayer. A better result can be observed if CoDel is applied. In this case, the bandwidth is also nearly fully utilized, and the amount of packets accumulated at the RLC buffer is reduced. On the other hand, DRQL, DynRLC and 5G-BDP present a very similar RLC buffer occupancy graph. The three of them try to dynamically adjust RLC buffer size to the radio link channel capacity (i.e., line in red). This ability enables them to come closer to the optimal queue size value where just enough packets to serve the PHY layer wait at the RLC buffer, but not more. Conversely, measuring the channel bandwidth at the senders congestion control algorithm is not as efficient as directly controlling the RLC queue size, as the BBR graph in Fig. 3.7 demonstrates.

In Fig. 3.3, we showed that there exist two inaccessible regions (i.e., minimum system response time and maximum utilization). The normalized utilization cannot surpass 1.0 and the minimum delay cannot go below 5 *ms* on average in our emulation scenario. The MAC scheduler pulls packets every 10 *ms* and, therefore, the optimal theoretical average time for an empty queue is 5 *ms*. These results come from the fact that the newly arrived packet may have to wait between [0, 10] *ms* before it is forwarded to the next sublayer. Therefore, in the best case, a packet would wait 0 *ms*, while in the worst case it would have to suffer a sojourn time of 10

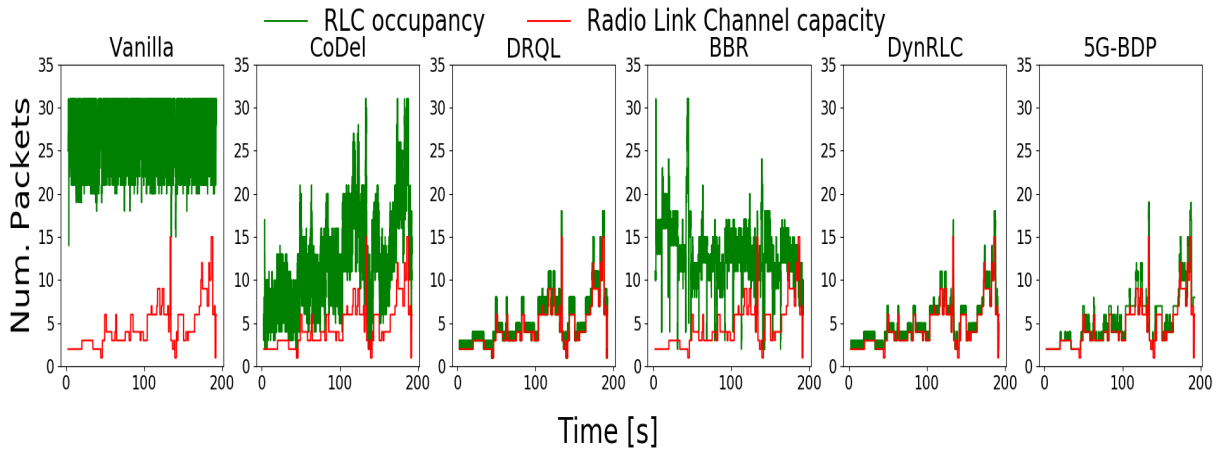


Figure 3.7: Radio link channel capacity and RLC queue occupancy.

$ms$  in an empty queue, being  $5 ms$  the average if a uniform distribution is assumed. The observations from Section 3.3 are shown in practice in Fig. 3.8 and Fig. 3.9. Both figures represent the empirical result of what we theoretically studied earlier in Section 3.3, where through a deterministic model, we reached the conclusion that no more data than the data accepted by the MAC should be kept at the RLC in any moment to work at the optimal  $\beta$  knee from the Fig. 3.3. From Fig. 3.8, 5G-BDP presents the best results as it maintains nearly full utilization (0.99947, 0.99949) and presents the lowest delay (9.12, 9.43)  $ms$ . The average delay is even below the  $10 ms$  value used as the TTI for this emulation scenario. This value shows that packets have a nearly free path until the RLC queue, and do not have to wait for a TTI on average. 5G-BDP achieves to work near the theoretical low limit of  $5 ms$  for this scenario, while maintaining near full utilization thanks to the RLC queue size dynamicity and the pacing algorithm. On the other hand, BBR does not work within the granularity requested by 5G's cellular network stack and, therefore, fails as a candidate for delay-sensitive communications. BBR, recalculates the available bandwidth every 10 seconds as it can be seen from the BBR peaks in Fig. 3.7, and adjusts the calculated bandwidth for the next 10 seconds. Such an approach lacks the ability to correctly adapt to the dynamic nature of 5G and, therefore, presents worse metrics than the other algorithms tested. BBR provides an utilization in the range of (0.98007, 0.98619) and a delay in the range of (33.40, 42.50)  $ms$  as it can be observed from Fig. 3.8. CoDel algorithm also exploits the communication between the sender and the receiver through TCP's congestion control algorithm discarding packets if congestion is detected. It needs some time until the congestion status information from the buffer can be transmitted to the sender through TCP's fast recovery mechanism. As seen in Fig. 3.8, it presents a normalized utilization that is in the range of (0.99967,

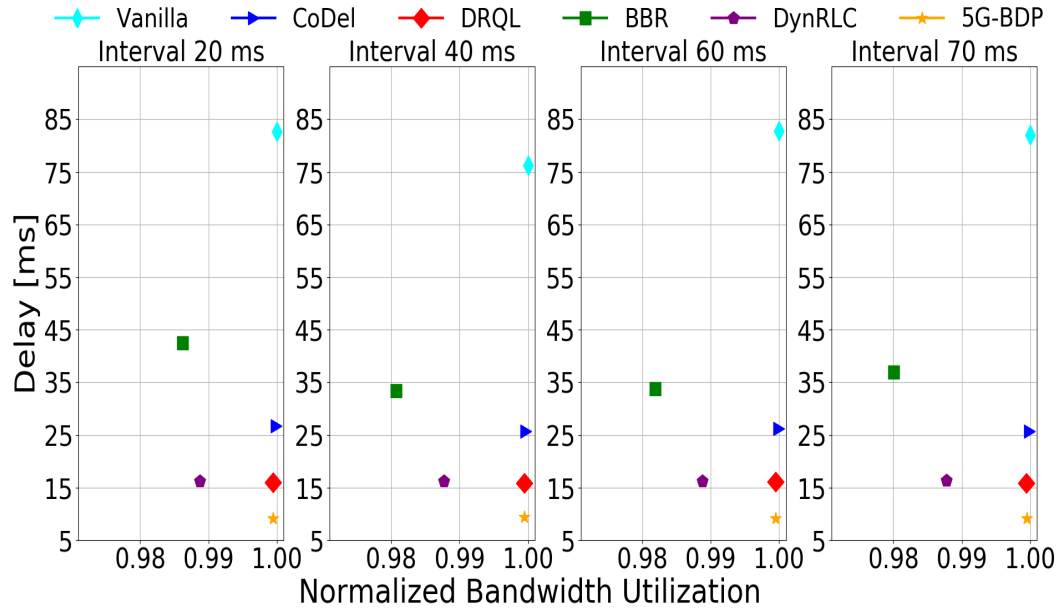


Figure 3.8: First scenario and pedestrian dataset: utilization rate and delay.

0.99999) at a cost of a delay in the range of  $(25.66, 26.69)$  ms. As demonstrated in the literature [68], CoDel works best if a little buffer is maintained after it, instead of collocating it directly at the edge. Since CoDel does not differentiate the packet type, 2.2% of the total low-latency packets were also discarded. Lastly, DRQL shrinks RLC's queue size in one step obtaining a normalized utilization range of  $(0.99994, 0.99999)$  (cf. Fig. 3.8) thus, slightly surpassing DynRLC that adapts its capacity through steps with a normalized utilization of  $(0.98774, 0.98876)$ . DRQL and DynRLC act directly on the queue without having to wait for additional information on transport delays, restraining the growth of the queue and ultimately reducing the delay. The delay ranges for DRQL and DynRLC in Fig. 3.8 are  $(15.92, 16.11)$  ms and  $(16.23, 16.40)$  ms, respectively. Thus using the pedestrian dataset, 5G-BDP reduces the delay in average by approximately 7.9 times with respect to Vanilla, 3.5 times compared to BBR, 2.7 times compared to CoDel and 1.7 times compared to DRQL and DynRLC. For the train dataset, the assessments from the pedestrian case hold as it can be observed in Fig. 3.9. Even though both datasets provide different radio channel link profiles, both represent real models, and, as shown in Fig. 3.9, the algorithms proposed also adapt to different circumstances efficiently.

In Fig. 3.10 and 3.11 the resulting delay values for the low-latency packets are presented with the lower and upper quartile values (i.e., 25<sup>th</sup> and 75<sup>th</sup> values). The horizontal line within the boxplots represents the median value, and the whiskers represent the range of the data. From



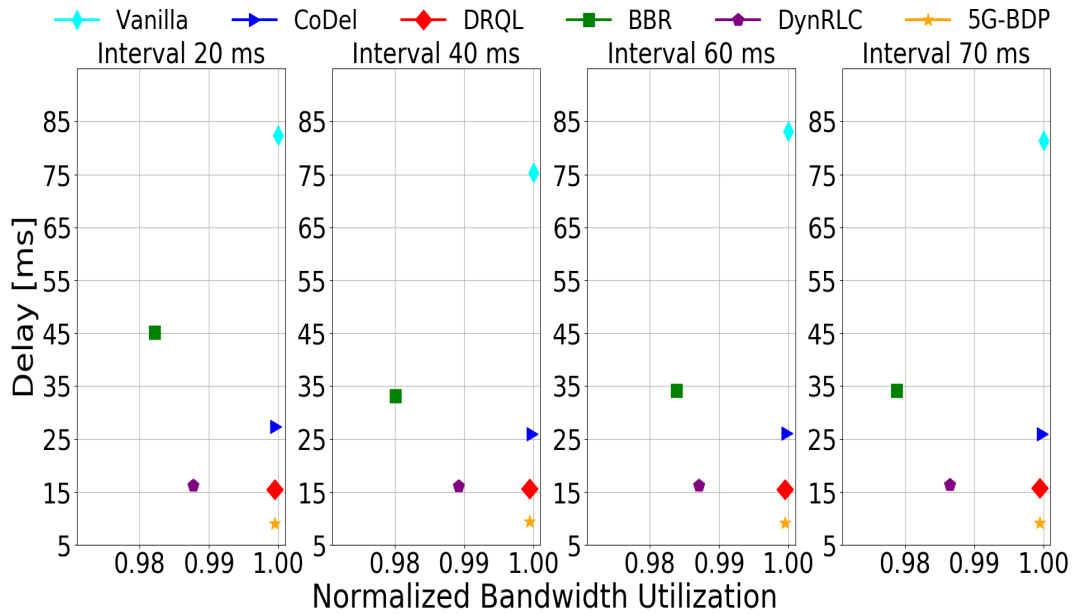


Figure 3.9: First scenario and train dataset: utilization rate and delay.

these figures, it can be extracted that the solutions that maintain the delay low, also present low jitter, which makes them appropriate for environments where not only the latency is considered, but also the variance of the latency is important.

Scalability is one of the most important aspects regarding any network and 5G is not an exception. Therefore, we tested the delay vs. the normalized bandwidth when two low-latency flows compete for accessing the resources with a greedy bulky TCP flow. Only two low-latency flows are used due to the emulator limitations as explained in Section 3.6.1. Both experiments were run with the pedestrian and the train datasets, so that the effectiveness of the proposed algorithms was validated. As it can be observed in Fig. 3.12 and 3.13, similar results are obtained. 5G-BDP outperforms the other algorithms (i.e., Vanilla, CoDel, DRQL, DynRLC and BBR), maintaining the delay low, while achieving very good bandwidth utilization results. Small differences can be observed regarding whether the pedestrian or the train dataset are used, which confirms the robustness of the algorithms presented. In any case, all the algorithms here presented scale well when different flows with different requirements compete to access the scarce resources. It is worth mentioning BBR's behavior, as it slightly wastes some transmission opportunities, when competing with a second low-latency flow.

In summary, in this subsection, Vanilla, CoDel, BBR, 5G-BDP, DRQL and DynRLC have been tested. Note that an approach with the successfully deployed FQ-CoDel [78] does not apply

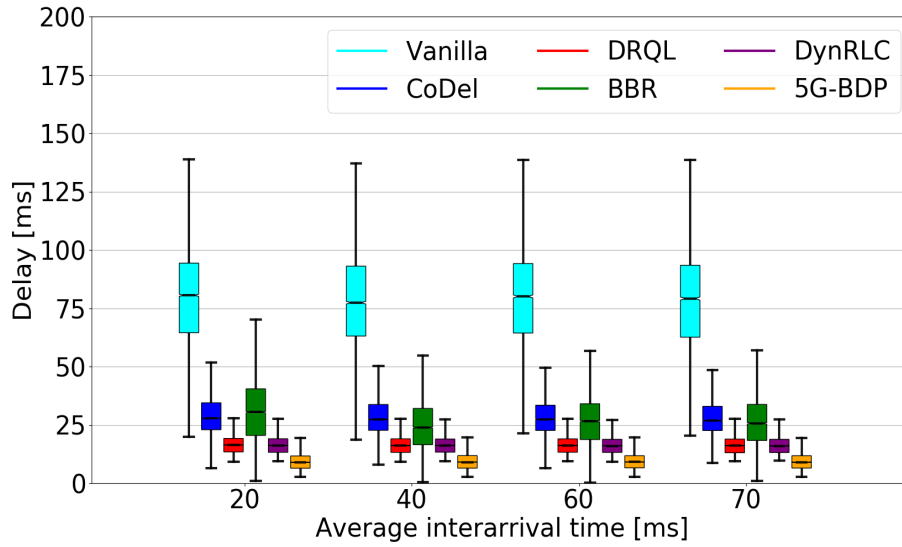


Figure 3.10: Low-latency traffic boxplot for pedestrian dataset and interval times of 20, 40, 60 and 70 ms.

in this scenario. The bufferbloat is formed at the RLC buffer. Therefore, if we apply a FQ-CoDel approach at the SDAP, packets will flow to the RLC and form there the bufferbloat as the SDAP scheduler is 10 times faster than the MAC scheduler. On the other side, even though very interesting idea, we did not investigate the approach of creating a second DRB and managing the MAC scheduler, since i) by far, the most common scenario is the one with one DRB, and ii) FQ-CoDel uses a modified DRR where byte fairness is desired. The second condition does not hold for cellular networks as fairness among RBs instead of bytes is normally demanded and different DRBs have different QoS requirements. However, the idea of creating a second DRB is proposed by L4S [50], and thus, the idea of investigating how a scheduler should behave in 5G and L4S remains open to the best of the author knowledge.

From the scenarios analyzed, it can be clearly seen that the Vanilla solution accumulates an excessive number of packets in the bottleneck link in both scenarios, which leads to large average sojourn delays in the [75, 85] ms range according to Fig. 3.8 and Fig. 3.9. CoDel partially alleviates the bufferbloat through its ability to communicate with the sender's congestion control algorithm discarding packets. However, CoDel's approach still results in considerable average sojourn times (i.e., approx. 25 ms as shown in Fig. 3.8 and 3.9), and squanders some transmission opportunities. The approaches that segregate the traffic offer better results, as the low-

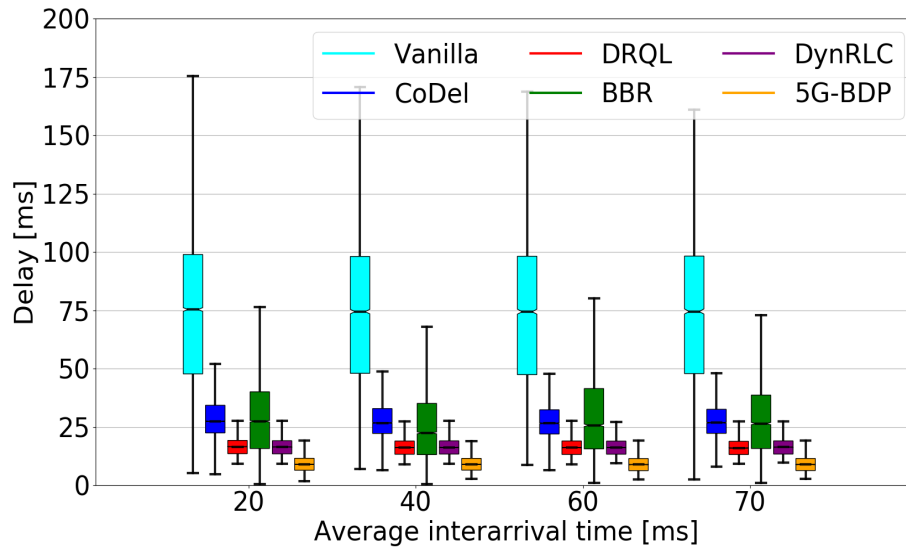


Figure 3.11: Low-latency traffic boxplot for train dataset and interval times of 20, 40, 60 and 70 ms.

latency traffic packets avoid the sojourn time at queues generated by greedy flows. Additionally, 5G-BDP's pacing capability allows it to avoid some sojourn time that, on the other hand, DRQL and DynRLC suffer from (i.e., low-latency packets with 5G-BDP experience delays of around 9 *ms* in average, against the around 16 *ms* in average of DynRLC and DRQL). DynRLC and DRQL lack of any pacing mechanism, and they forward packets at the beginning of the TTI, while 5G-BDP uniformly distributes the forwarding packets' process from the SDAP to the RLC in a TTI, thus, reducing the sojourn time for packets that arrive in the middle of a TTI. Moreover, DRQL and 5G-BDP, achieve the lowest standard deviation as shown in the results, being more stable in terms of jitter. All the methods tested present good resource utilization percentages (i.e., in the [0.95,1.0) range in accordance with Figs. 3.8, 3.9, 3.12 and 3.13). Both scenarios confirm the validity of the algorithms.

### Second Scenario: Shared SDAP Buffer, Shared RLC Buffer

For the second scenario, as shown in Fig. 3.5, the congestion avoidance is more complex as two levels of queues must be considered, and queues at upper entities (i.e., UPF) have side effects on queues at lower entities (i.e., SDAP and RLC). The RLC buffer must have enough bytes to forward once the MAC scheduler requests for transmission, to avoid wasting RBs, but keeping additional bytes just adds delay. Moreover, the SDAP queues should have just enough bytes to

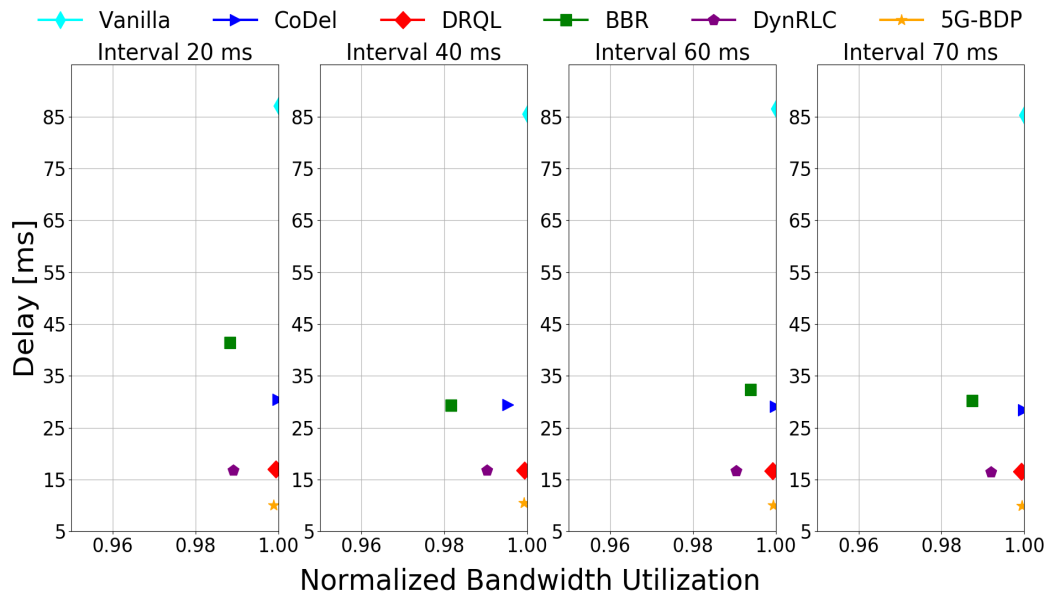


Figure 3.12: Delay vs. normalized bandwidth with two low-latency flows and a bulky TCP flow with pedestrian dataset.

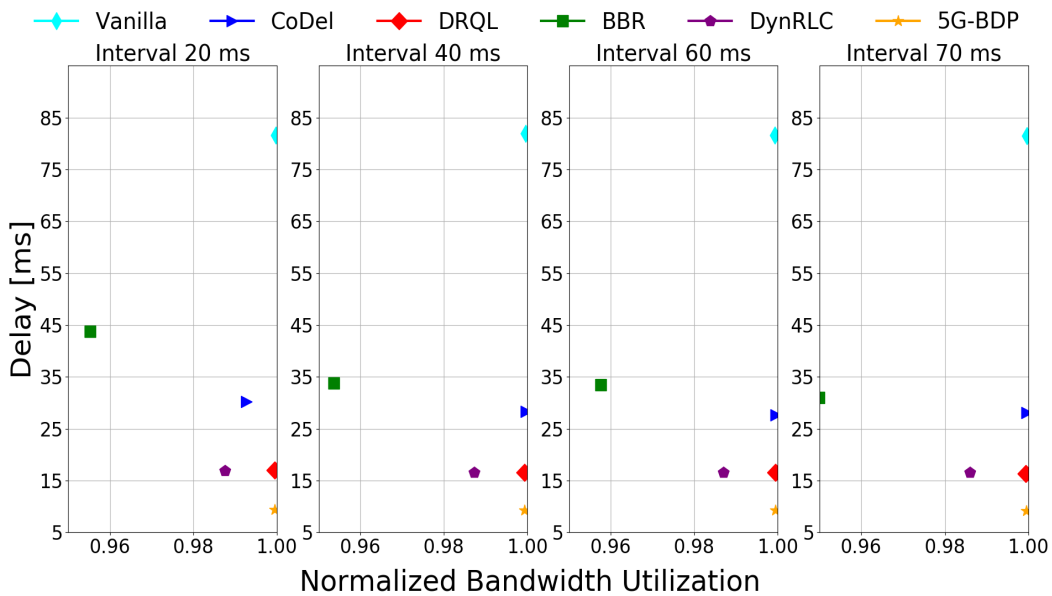


Figure 3.13: Delay vs. normalized bandwidth with two low-latency flows and a bulky TCP flow with train dataset.

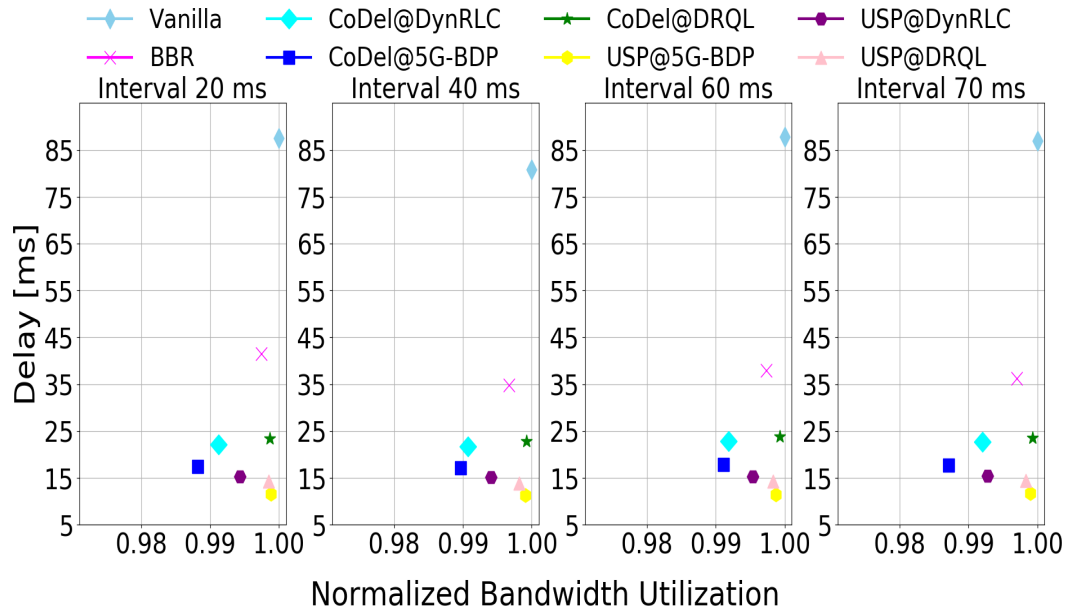


Figure 3.14: Second scenario and pedestrian dataset: utilization rate and delay.

feed the RLC buffer, but not more, while keeping the other packets at higher layers due to 5G's QoS funnel nature as seen in Fig. 2.3 and 2.4. The principle exposed by Kleinrock [25], which describes how data should be delivered at bottleneck's bandwidth speed to avoid starving the buffer and preventing forming queues, equally applies for the SDAP queues. From Fig. 3.14 and 3.15, it can be observed that the availability of a synchronous communication between the UPF and the SDAP is critical. The three methods tied with CoDel at SDAP queues present significantly worse results in delay, for both, train and pedestrian traces. It also shows that USP successfully feeds the SDAP queue with enough data to avoid a possible starvation at the RLC queue. From the moment where CoDel discards a packet due to excessive sojourn time until TCP's congestion control algorithm realizes that a packet was lost, the sender will send packets in a bursty manner. Even though such a solution may appear slow, it surpasses the BBR algorithm, where once again it clearly shows the limitations of tackling the bufferbloat problem in low-latency systems relying on the congestion control algorithm. The same argument applies to the TCP's congestion control algorithms that address the bufferbloat mentioned at Section 3.1.

Again, the combination of USP with 5G-BDP as seen in Fig. 3.14 and 3.15 (i.e., USP@5G-BDP) outperforms all other alternatives due to its pacing nature in comparison with the bursty nature of USP with DynRLC, or USP with DRQL. In particular, it approximately reduces the average delay by 7.3, 3.1, 1.9, 1.5, 2.1, 1.4, 1.3 times in comparison with Vanilla, BBR, CoDel@DynRLC,

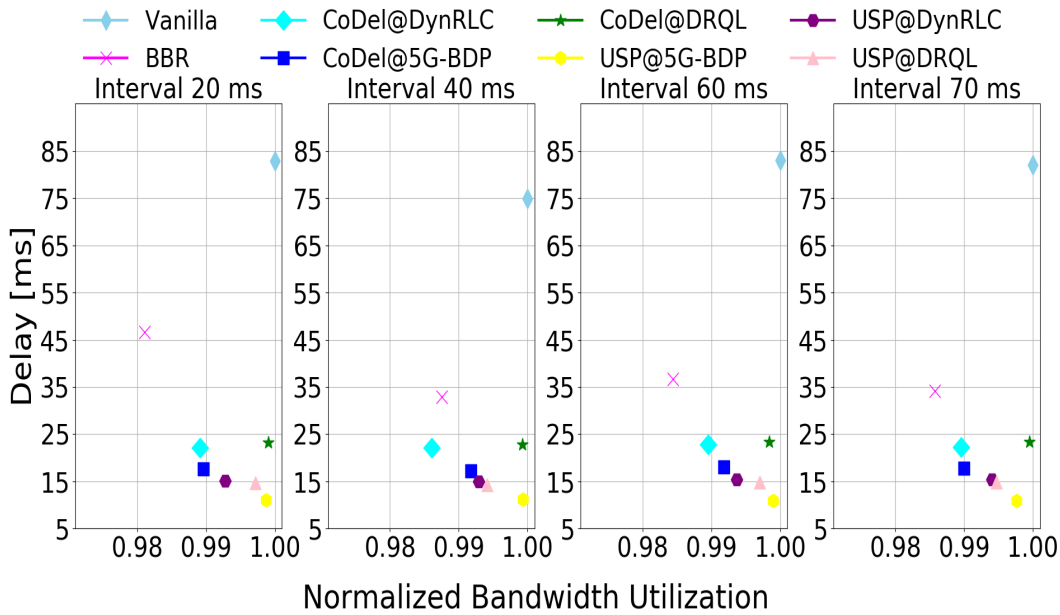


Figure 3.15: Second scenario and train dataset: utilization rate and delay.

CoDel@5G-BDP, CoDel@DRQL, USP@DynRLC and USP@DRQL, respectively. CoDel@DynRLC, CoDel@5G-BDP and CoDel@DRQL dropped 2.4%, 3.8% and 2.5% of the total low-latency packets, respectively. A pacing algorithm such as 5G-BDP, maintains the packets during larger time in the queue SDAP in comparison with bursty algorithms such as DynRLC or DRQL.

Therefore, in the CoDel and 5G-BDP combination (i.e., CoDel@5G-BDP) the packets dropped are slightly superior to the CoDel and DRQL (i.e., CoDel@DRQL) or CoDel and DynRLC (i.e., CoDel@DynRLC) combinations as CoDel interprets the sojourn time as a congestion symptom, and thus, the utilization is slightly inferior.

Regarding the scalability of the algorithms, Fig. 3.16 and 3.17 show the results when two low-latency flows and a bulky TCP flow compete for acquiring the resources. The same conclusions exposed before hold, where very similar results to Fig. 3.14 and 3.15 are obtained. However, note that the scale at x axis of Fig. 3.16 and 3.17 with respect to Fig. 3.14 and 3.15 changes, in order to depict the results of BBR. BBR, exactly as occurs in the first scenario when the scalability is tested (i.e., Fig. 3.12 and 3.13), presents some marginal bandwidth lost. The problem of dealing with the bufferbloat problem from the TCP's congestion control is again here explicitly exposed, where the delay of transmitting the congestion information until the sender can lead to resource underutilization.

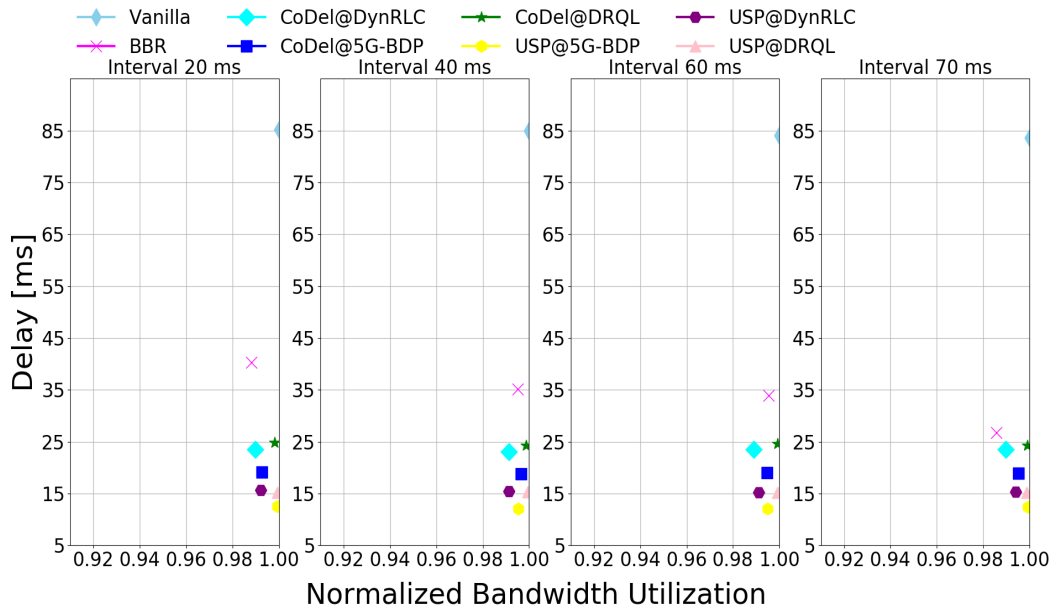


Figure 3.16: Scalability: Delay vs. normalized bandwidth with two low-latency flows and a bulky TCP flow with pedestrian dataset

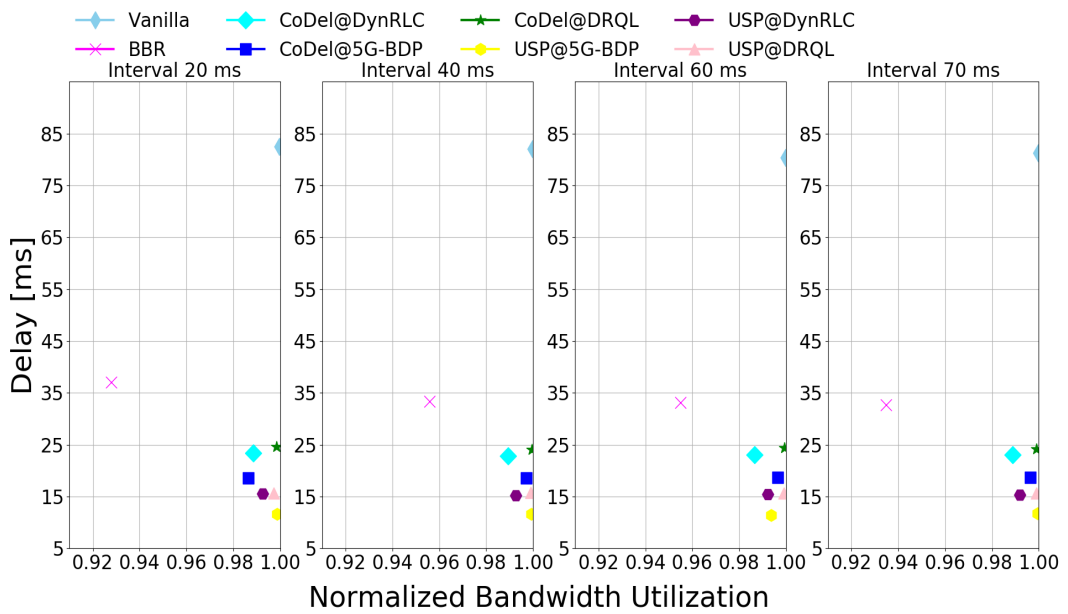


Figure 3.17: Scalability: Delay vs. normalized Bandwidth with two low-latency flows and a bulky TCP flow with train dataset

The results from this scenario, where the queues of a higher entity are involved (i.e., UPF), show worse performance metrics than the results obtained from the first scenario (i.e., individual SDAP buffer, shared RLC buffer vs shared SDAP buffer, shared RLC buffer). This is an expected result as, in the second scenario, three entities/sublayers queues (i.e., UPF, SDAP and RLC) need to be managed, while in the earlier scenario, two sublayers were managed (i.e., SDAP and RLC). In essence, in this second scenario less resources are acquired (i.e., one buffer at SDAP in contrast with two buffers at the first scenario), and therefore, the flows share longer paths which ultimately increments the delay. Fig. 3.16 and 3.17 show that adding a second low-latency flow does not alter the aforementioned conclusions qualitatively.

### 3.6.3 Cellular Network Testbed: OAI

Once the algorithms were validated in the emulator, the next natural step required implementing them in a testbed. Fig. 3.18 illustrates the implementation. A B-200 Ettus Universal Software Radio Peripheral (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 in the eNodeB side. Two commercial UEs were used. For the first UE, a Huawei E3372 LTE Universal Serial Bus (USB) stick connected to a Raspberry Pi 4 Model B was used. For the second UE, a 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. To implement the testbed we use OAI.

OAI is a “*non-profit consortium fostering a community of industrial as well as research contributors for open source software and hardware development for the core network, access network and user equipment of 3GPP cellular networks [79]*” according to their web page. They provide one of the most advanced open source cellular network stacks, recently having achieved the connection with a 5G OPPO mobile<sup>5</sup>. Unfortunately, OAI still needs some major development to be able to connect and use a large subset of 5G’s capabilities. However, the RLC sublayer does not significantly change and the results obtained with the LTE stack plus the mapping and buffering capabilities at the SDAP sublayer are still relevant. Therefore, OAI was selected as the 3GPP compliant cellular network stack where the algorithms: Vanilla, CoDel, DRQL, BBR, DynRLC and e5G-BDP were tested. OAI was enhanced with the SDAP sublayer, where a queue per QFI and UE was

---

<sup>5</sup>The current OAI implementation subset of the 3GPP stack can be accessed at [12] and was presented at [80].



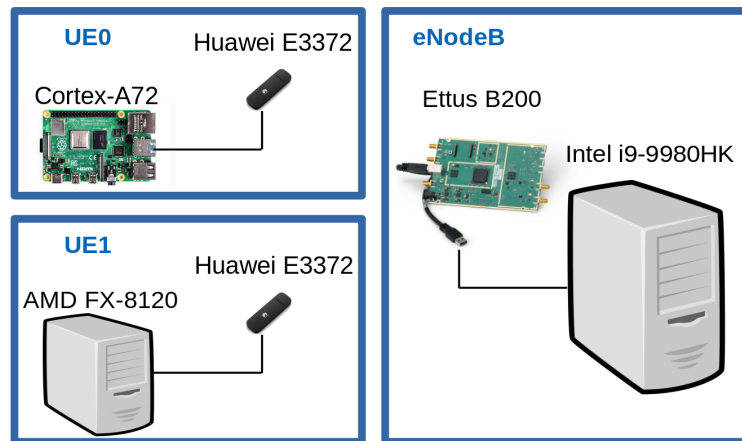


Figure 3.18: Evaluation testbed.

implemented.

Data packets arrive into the newly developed SDAP sublayer, where a traffic classifier segregates the traffic according to the 5-tuple (i.e., IP source/destination address, ports and protocol), imitating the QFI marking, and forwards them to a statically allocated QFI queue. A pacifier, considering the aggregated traffic of all the queues, determines whether to forward a packet to PDCP or maintain it. We implemented a priority policy, where the next packet considered to dequeue by the scheduler is selected among the highest priority non-empty queues. Packets are lastly gathered at the RLC buffers as RLC SDUs, where they wait until the MAC scheduler requests a number of bytes from a queue, and the RLC sublayer, if necessary, segments the SDUs to form a PDU that fulfills the required bytes and forwards them.

To create a realistic scenario where traffic flows with different requirements co-exist, a flow with low-latency requirements (e.g., a VoIP flow) and a competing bandwidth driven flow (e.g., an application software update) have been generated per UE, unless otherwise indicated. The bandwidth driven flow has been generated using the *iperf3* tool with TCP traffic and a maximum segment size (i.e., MTU - 40 bytes) of 1460 bytes. For the low-latency driven flow, the Isochronous Round-Trip Tester (*irtt*) [81] has been used, where a G.711 VoIP conversation is emulated through UDP data frames of 172 bytes with an interval of 20 ms [74], resulting in a bandwidth consumption of 64 Kbps. Both flows are segregated at the SDAP sublayer according to the 5-tuple and the UDP flow's packets are placed in a higher priority queue. TCP Cubic [14] is used for all the bandwidth driven flows as it is the most released TCP congestion control

protocol, as well as the default in the tested Linux kernel<sup>6</sup>, except when BBR [22] is explicitly mentioned. Due to the dynamic nature of the radio link channel, the UE reports the CQI in the uplink to the eNodeB so that a MCS, where the data transmission error probability drops below 10% as specified by 3GPP, is selected [82]. The MCS index approximately determines the amount of bytes that a RB transmits, and therefore, the available bandwidth (i.e., number of bytes / TTI) in the next TTI (i.e., 1 ms as OAI implements LTE's TTI). We evaluated 3 different MCS scenarios. In the first one, OAI selects the MCS according to the UE's reported CQI. In the second and third scenarios, the UE's reported CQI is ignored, and the MCS is selected according to the CQI reported by real LTE traces [26] from the pedestrian and the train UE dataset described in Section 3.6. In this manner, we expose our algorithm to more realistic, yet diverse MCS profiles. This technique can be applied as the real channel is better than the emulated one. Note that if the real channel would be worst than the emulated one, the HARQ MAC mechanism would be continuously trigger, and the throughput would be affected. As previously mentioned, the cellular network is exposed to services with heterogeneous QoS requirements, that can contain several flows each. To this end, we also evaluated 1, 2, 4, and 8 VoIP flows in parallel, to emulate a multi user VoIP call. To generate the bufferbloat, the bulky traffic starts 5 seconds before the VoIP flow. The VoIP lasts for 60 seconds, which represents 60000 TTIs (i.e.,  $60000 \text{ ms} \times 1 \text{ TTI/ms}$ ) in which 3000 UDP packets are sent. Every test has been repeated 5 times (i.e., 300000 TTIs or 5 minutes) to correctly verify the results and minimize the possible outliers' effect. To report the delay, a timestamp is added to all the packets in an external data structure once they enter the SDAP sublayer.

The elapsed time is measured in the VoIP packets once they are forwarded from the RLC buffer to the MAC sublayer. In this manner, the delay generated by the bufferbloat can be precisely isolated, studied and its impact quantified. The default bearer has been mapped to a RLC UM bearer, as it was OAI's default option. It has a hard capacity limit of  $5 \times 10^6$  bytes<sup>7</sup>, which suffices to bloat the 5MHz eNodeB RLC buffer with a 28 MCS index for  $5 \times 10^6 \text{ bytes} / (2289 \text{ bytes} / 1 \text{ ms}) \approx 2.5 \text{ seconds}$ . A 25 RBs configuration (i.e., 5 MHz) was selected, which reaches a maximum downlink throughput of approximately 16-17 Mbits/sec [12]. The achieved bandwidth is measured by the *iperf3* tool (i.e., measuring the TCP packets that arrive to the UE), as well as count-

---

<sup>6</sup>Linux kernel 5.3.0-62-lowlatency

<sup>7</sup>File `openair2/LAYER2/RLC/r1c.c`

ing the unused transmission opportunities that happen every TTI at the eNodeB (i.e., unfilled RBs), and thus, our results are doubly checked.

The Vanilla case represents the traditional approach taken in cellular networks. It is OAI's default behavior. For the evaluation of CoDel, we substituted the default RLC FIFO buffer with a CoDel queue according to the values described in [41] with the *interval value* is set to 100 *ms* and *target value* is set to 5 *ms*. For testing BBR, we used the default BBR algorithm provided by the Linux 5.3.0-62-lowlatency kernel.

Lastly, the 3 algorithms that permit a rapid low-latency packet delivery from within the cellular network are evaluated: DynRLC, DRQL and e5G-BDP. DynRLC measures the sojourn time suffered by the SDUs at the RLC and adjusts the maximum number of allowed SDUs in the RLC buffer consequently. The sojourn time is captured for every packet that leaves 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  $\alpha$ , marks the increase or reduction rate. The new maximum number of allowed SDUs occur every interval of time [23], which we set to 10 *ms* or a radio frame in LTE, in contrast to the 200 *ms* mentioned by [23]. The target delay is set to 3 *ms* instead of the minimum of 30 *ms* reported by [23], as a smaller value was heuristically found not to achieve full bandwidth, and a bigger value just adds unnecessary sojourn time. The maximum number of packets is set to 10, a high enough value for our test scenarios (i.e.,  $10 \times MTU = 15000$  bytes, which is a much greater value than the maximum number of bytes per TTI in a 5 MHz eNode, which is 2289 bytes), and the minimum value to 1. The  $\alpha$  value is set to 0.05, so that in less than 500 *ms* or 50 interval of times, the packet number can change from 1 to 10 (i.e.,  $1 \times 1.05^{500/10} > 10$ ).

DRQL [73], on the contrary, is based on the number of bytes that remain at the RLC buffer after a TTI. Therefore, it also requires a cross-layer communication between the RLC and the SDAP sublayers. No parameters need to be adjusted as DRQL tries to maintain the equivalent of one MTU in the RLC buffer after the TTI. 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 possesses a pacing mechanism that augments the probabilities of delivering a packet within a TTI. It is also a knob-less mechanism as no parameters need to be configured.

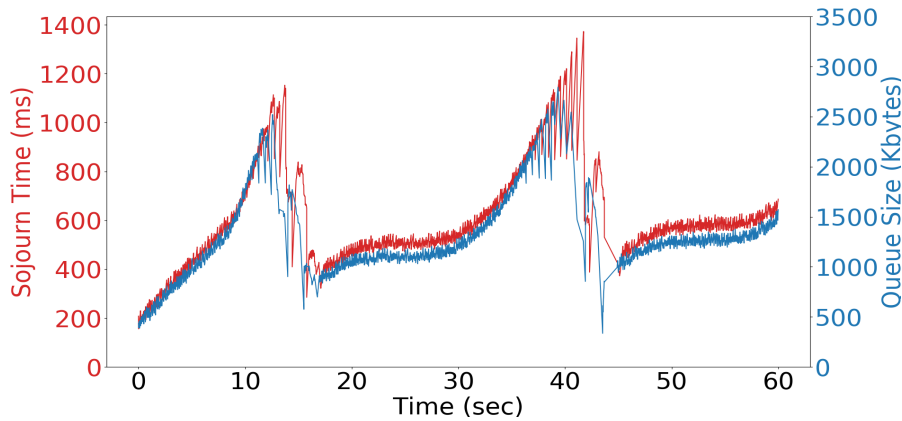


Figure 3.19: Delay suffered by a VoIP flow when sharing the RLC buffer with a bulky flow in Vanilla OAI.

### 3.6.4 Cellular Network Testbed's Experimental Results

3GPP standard does not include any mechanism capable of tackling the bufferbloat problem in current cellular networks. Therefore, its stack is susceptible to experience considerable delays, as it is equipped with large buffers and generally the RAN represents the slowest link in the data path. Thus, packets with low-latency requirements that share the RLC buffer with bulky flows will suffer from large buffer depletion time. Such effect can be clearly seen in Fig. 3.19, where the correlation between the VoIP packet delay and the RLC queue size can be clearly observed for the Vanilla deployment, where the packets sojourn time at RLC is the governing delay factor. In Vanilla deployments, the delay for low-latency flows can grow up to the order of seconds as reported by [15] and shown in Fig. 3.19.

Remind that the RLC buffer should contain just enough bytes to satisfy the MAC sublayer demands. Any additional byte just augments the delay [73], 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 and the non-uniform arrival pace of packets.

In Fig. 3.20, the RLC buffer occupancy of the six different methods is shown. Note the change of scale between the different solutions. It can be observed that CoDel and BBR reduce the amount of bytes at the RLC queue by at least 5 times when compared with Vanilla. However,

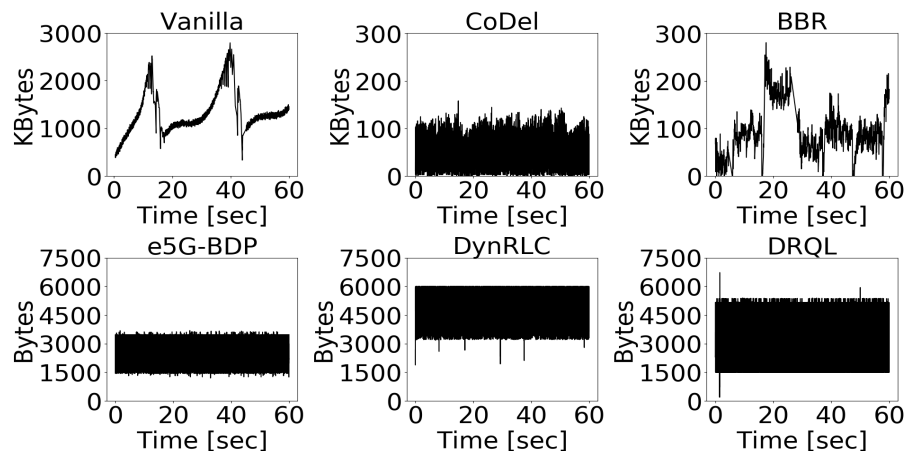


Figure 3.20: RLC buffer occupancy for different methods for MCS index of 28 and 25 RBs.

CoDel’s mechanism for discarding packets results in transmission opportunity losses. On the other hand, BBR periodically drains the bottleneck queue after 10 seconds during 200 *ms* if the measured RTT does not decrease during that period [22], leading to a small bandwidth reduction. This effect can be clearly observed in Fig. 3.20, as the BBR RLC buffer gets 6 times drained during the 60 seconds. However, the aforementioned three methods (i.e., Vanilla, CoDel and BBR) create a considerable 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 bandwidth, 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 that shares the bottleneck buffer (i.e., RLC buffer) with bulky traffic flows will suffer less queuing depletion time, as the amount of bytes there is meticulously maintained low.

As observed in Fig. 3.20, e5G-BDP is able to maintain the 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. 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 IP version 4 (IPv4) header and 8 bytes of UDP header). Therefore, with a MCS index of 28, the optimal queue occupancy should never exceed 2289 bytes. However due to the packet sizes, achieving such objective may not be possible. In any case, the buffer should contain at least 2289 bytes when a notification from the MAC sublayer arrives, not to waste a transmission opportunity, but any additional byte just adds

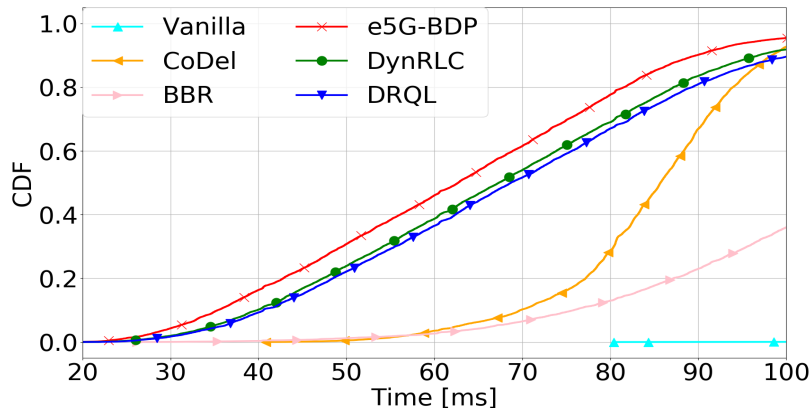


Figure 3.21: Cumulative Distribution Function (CDF) of VoIP flow's RTT.

delay. Hence, in Fig. 3.20 it is shown that the algorithms that maintain the queue occupancy low, are ultimately the algorithms that achieve lower delays.

In Fig. 3.21, the same conclusions from the RTT perspective reported by *irtt* are explicitly drawn. The RTT is composed by the stack delay (i.e., uplink and downlink procedure), the protocol delay (e.g., uplink Scheduling Request), the packet routing and various buffering delays (e.g., NIC buffers). However, the importance of segregation in the downlink procedure is clearly shown in Fig. 3.21, 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. It can be also concluded that in our cellular network testbed a minimum of 20 *ms* RTT delay exists.

As it can be observed in Fig. 3.22, packet delivery under Vanilla, CoDel and BBR face an order of magnitude larger delay than e5G-BDP, DynRLC or DRQL. In fact, the time where their CDF asymptotically converge into 1.0 is located way after the scope of Fig. 3.22. Note that in the cases of Vanilla, CoDel and BBR, the SDAP acts just as a mapper, it adds latency orders of magnitude smaller than the RLC buffer, and therefore, no graph is depicted in the second row of the Fig. 3.22. Between e5G-BDP, DRQL and DynRLC, the lack of a pacing mechanisms in DRQL and DynRLC has notorious effects. Every new TTI, the SDAP scheduler policy prioritizes the low-latency packets against the bulky traffic ones. This effect can be clearly seen in Fig. 3.22 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 packets at RLC wait between (750, 1000)  $\mu s$ , meaning that they were forwarded from

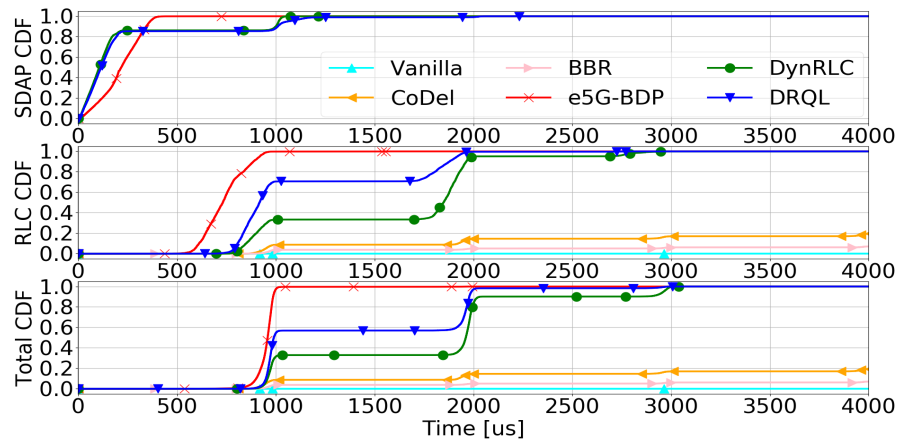


Figure 3.22: CDF of the queuing delay at RLC, SDAP and the sum of both for a MCS index of 28 and 25 RBs.

the SDAP sublayer in the  $(0, 250)$   $\mu s$  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 occurs (i.e., an additional delay of  $1000 \mu s$  is added at the SDAP sublayer for DRQL and DynRLC). On the other hand, the pacing capabilities of e5G-BDP permits that a newly arrived packet at the SDAP is forwarded to the RLC immediately, avoiding to have to wait for the next TTI. Such a behavior is also appreciable at the RLC queue, where the e5G-BDP permits ingressing packets at any time within a TTI, considerably reducing the delays at the RLC buffer (i.e., the packets stay at the RLC sublayer between  $(500, 1000)$   $\mu s$ ). This is the main reason that allows e5G-BDP to avoid waiting for a new TTI, and therefore, outperforms DRQL or DynRLC as shown in Fig. 3.22, 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  $1000 \mu s$ .

The throughput has been quantified from two perspectives. First we measured the results received from *iperf3* (i.e., the bulky flow) that rely on Layer 4 measurements. The results can be observed in Fig. 3.23, where Vanilla, CoDel, BBR, e5G-BDP, DynRLC and DRQL reported 16.58, 12.95, 16.05, 16.20, 14.35 and 16.30 MBits/sec, respectively. Then, we quantified the number of RBs that are not used during the 60 second VoIP conversation. The Vanilla case maps data to all the RBs during all the 60 seconds and therefore has the highest achievable throughput. BBR, as shown in Fig. 3.20, drains the buffer in order 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

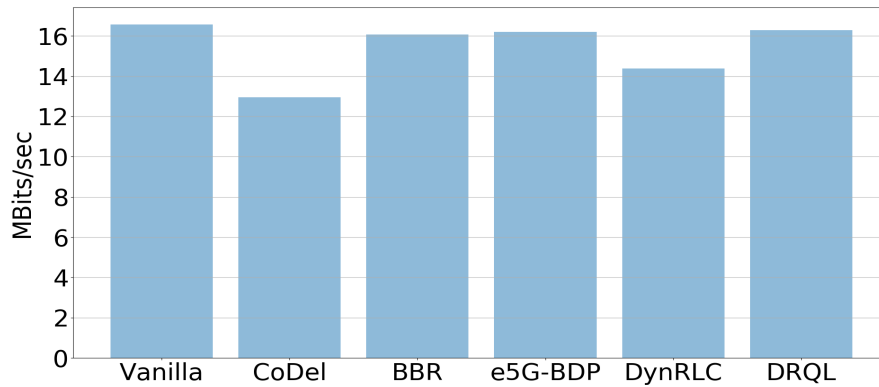


Figure 3.23: Bandwidth reported by *iperf3* with MCS=28 and 25 RBs.

60 seconds of the 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, respectively. Lastly, e5G-BDP also loses a 1.9% of transmission opportunities. These results match the outcomes provided by *iperf3* as shown in Fig. 3.23. 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  $\mu s$ ) as seen in Fig. 3.22, while its direct competitors (i.e., DynRLC and DRQL) need at least two TTIs (i.e., 2000  $\mu s$ ) for this percentage of packets.

To measure the scalability, the effect of 1, 2, 4 and 8 VoIP connections has been tested. DynRLC relies on the number of SDUs in the RLC and the delay they suffer in order to adjust the amount of SDUs at the buffer. However, the MAC sublayer assigns RBs to the RLC buffer (i.e., this can be translated into bytes) rather than SDUs. This creates a weakness on DynRLC when packets of different sizes are mixed (i.e., DynRLC considers equally a SDU of 1500 or 200 bytes), as DynRLC calculates the optimal number of RLC SDUs. For example, DynRLC can estimate the optimal number of RLC SDUs to be 4, but 4 low-latency RLC SDUs (i.e.,  $4 \times 200 = 800$  bytes) greatly differs from 4 bulky RLC SDUs (i.e.,  $4 \times 1500 = 6000$  bytes). Therefore, DynRLC wastes transmission opportunities (i.e., not containing enough bytes at the RLC when the MAC requests them) and creates 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). On the other hand, e5G-BDP and DRQL are methods that do not rely on the amount of SDUs but rather on the number of bytes, resulting in more robust algorithms under the scenario described, as



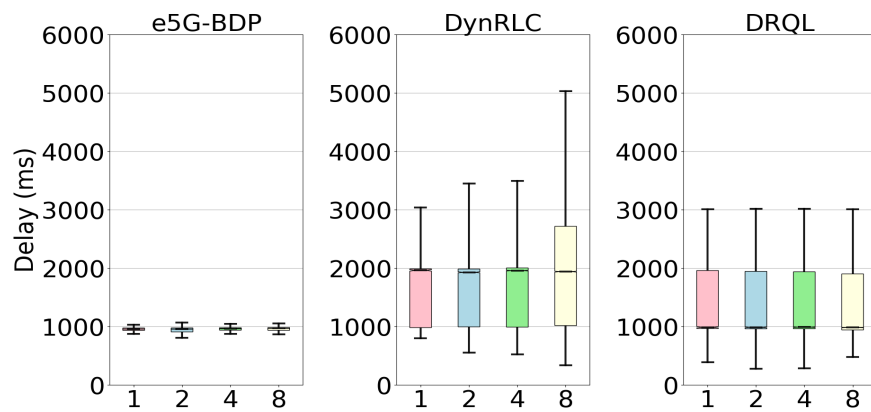


Figure 3.24: Delay faced by VoIP packets with 1, 2, 4 and 8 flows.

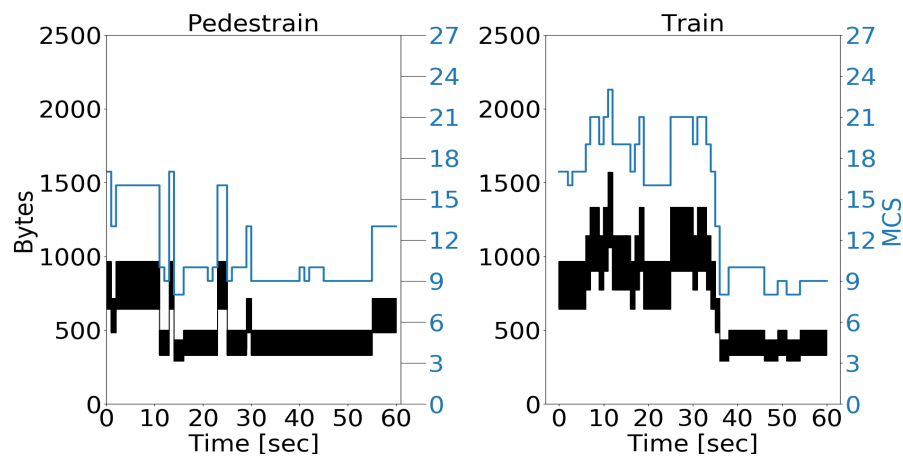


Figure 3.25: Amount of bytes requested by the MAC sublayer and MCS employed.

shown in Fig. 3.24.

The previously shown results have been obtained in a controlled static environment with the MCS index reported by the UE and nearly no interference. However, in real cellular network deployments, the radio link capacity can abruptly change [26]. Therefore, we emulated such scenario setting the MCS index following real LTE traces [26]. We chose the same scenarios as the simulated ones (i.e., pedestrian and train scenarios) with different MCS profiles. In Fig. 3.25, we show the MCS values from these two traces and the number of bytes requested by the MAC sublayer during the 60 seconds of the VoIP conversation. As seen in Fig. 3.25, they are highly correlated, and therefore, a MCS reduction directly affects the available bandwidth during the following TTI (i.e.,  $\text{bandwidth} = \text{bytes requested} / \text{TTI}$ ). Moreover, the sudden MCS variation at around the 35<sup>th</sup> second in the train scenario presents a challenge for the algorithms

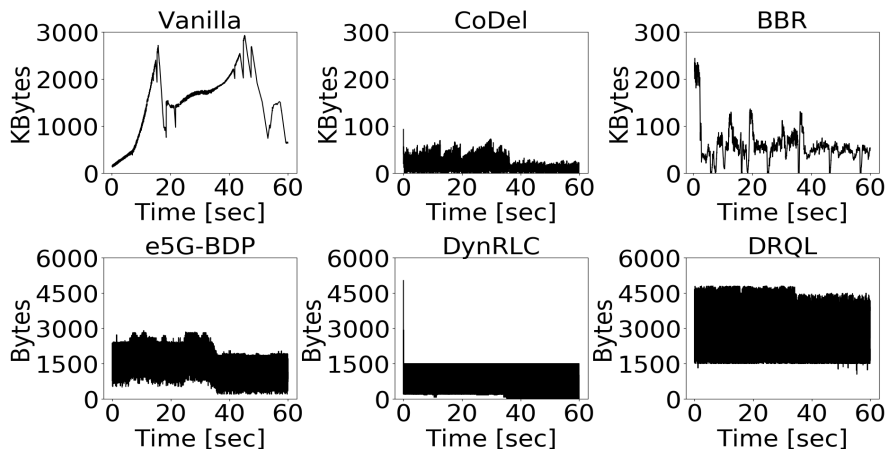


Figure 3.26: RLC buffer occupancy for different methods for train dataset and 25 RBs.

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. 3.26. A reduction at around the 35<sup>th</sup> second in the amount of bytes at the RLC buffer can be observed for CoDel, BBR and e5G-BDP, while no specific change at Vanilla, DynRLC or DRQL can be seen. Fig. 3.26 clearly shows DynRLC limitations, when the available bandwidth is reduced. In this dynamic scenario e5G-BDP still delivers the low-latency packets faster than DynRLC as it can be observed from Fig. 3.27. No algorithm is capable of maintaining full bandwidth and forwarding low-latency packets within a TTI. Such outcome results from the fact that no preemption is allowed from the RLC queue in 3GPP's described model. Therefore, if a packet of 1500 bytes has already been forwarded to the RLC and just 500 bytes are pulled every TTI, the low-latency packet will suffer the depletion time from the previous packet (i.e., at least 3 TTIs or 3000  $\mu s$  in LTE networks).

The average throughput for the train scenario reported by *iperf3* can be observed in Fig. 3.28, 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, respectively.

One of the most important features when analyzing the bufferbloat problem is the bandwidth vs. delay dichotomy. 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 can be formed, as explained in Section 3.3. However, achieving both is very challenging as not only

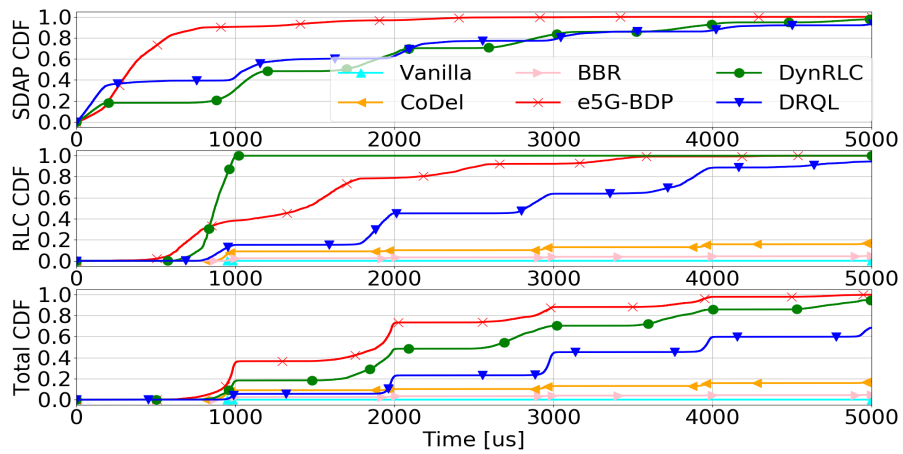


Figure 3.27: CDF for train scenario and 25 RBs.

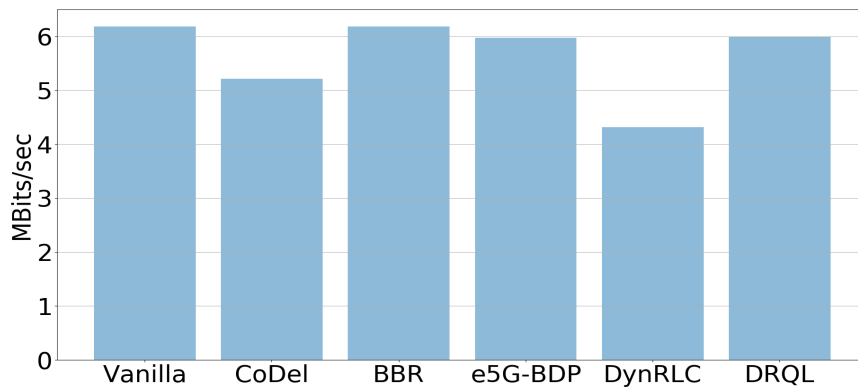


Figure 3.28: Bandwidth for the train scenario.

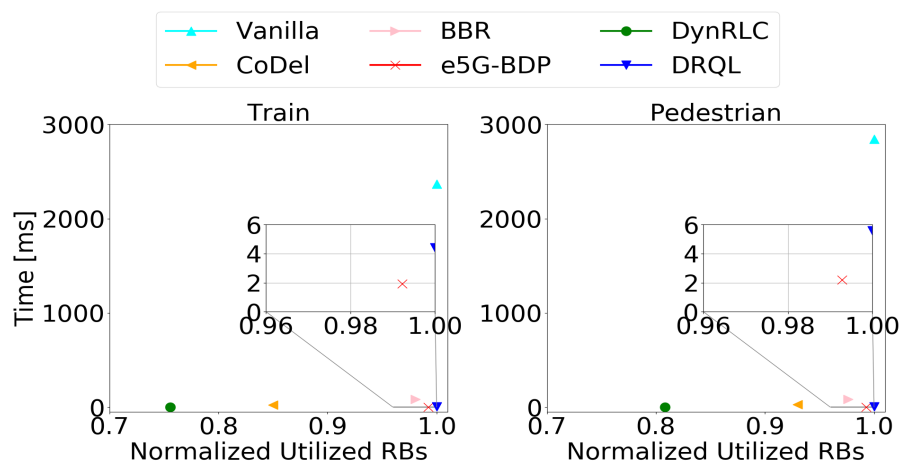


Figure 3.29: Normalized bandwidth vs average delay for train and pedestrian scenario.

the varying cellular network radio link effects have to be taken into account, but also TCP's behavior has to be considered. As shown in Fig. 3.29 (which is the practical result of the theoretical Fig. 3.3), the only solutions that maintain a nearly 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. Table 3.2 shows numerically the average delay values presented in Fig. 3.29. The solutions based on the cellular network 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 actual bufferbloat problem in cellular networks. Table 3.3 depicts the 95% CDF value for average delay as shown in Fig. 3.27, while Table 3.4 displays the percentage of used RBs in the train and pedestrian scenario. It can be observed that in both scenarios, e5G-BDP outperforms in average every competitor in sojourn time by at least 2 TTIs, while squandering less than 1% of the available resources, and therefore, is the best solution tested for bufferbloat avoidance at the RLC buffer.

Table 3.2: Average Delay in ms at eNodeB for the train and pedestrian scenarios.

	Vanilla	CoDel	BBR	e5G-BDP	DynRLC	DRQL
Train.	2366.08	24.27	84.06	<b>1.93</b>	2.67	4.42
Ped.	2841.73	31.58	85.60	<b>2.19</b>	2.92	5.59

Table 3.3: 95% Delay in ms at eNodeB for the train and pedestrian scenarios.

	Vanilla	CoDel	BBR	e5G-BDP	DynRLC	DRQL
Train.	5325.93	52.93	186.94	<b>3.93</b>	5.56	8.94
Ped.	5261.16	138.94	218.93	<b>3.90</b>	5.57	8.77

Table 3.4: Percentage of used RBs for the train and pedestrian scenarios.

	Vanilla	CoDel	BBR	e5G-BDP	DynRLC	DRQL
Train	<b>100%</b>	85.0%	98.1%	99.1%	75.6%	100%
Ped.	<b>100%</b>	93.0%	97.6%	99.3%	80.9%	100%

### 3.7 Summary

In this Chapter 3, the bufferbloat problem in the 5G stack has been exhaustively studied. In the first part, a survey with current state-of-the-art solutions in 5G have been presented, and their limitations and achievements exposed. Different solutions with 5G stack's specificities that emerge from diverse paradigms (i.e., Linux traffic control mechanisms and queuing theory) have been presented, and novel algorithms have been proposed. Specifically, Kleinrock's paper [25] has been considered, and the *keep the pipe full, but not fuller* principle with 5G specificities applied. (e)5G-BDP and USP are the concrete solutions from such theoretical analysis, and have been firstly implemented in a self developed 5G QoS hierarchical queuing emulator, and have been later ported to OAI and tested with COTS hardware. The experimental outcomes from the emulator, as well as the testbed results have been presented. They both confirm the validity of the algorithms.

From the bufferbloat perspective, the following conclusions can be outlined:

- 3GPP's proposed 5G stack is susceptible of generating large buffers at the RLC buffer (i.e., the bottleneck), which can cause delays in the order of seconds.
- If the bufferbloat is to be avoided, 3GPP should explicitly define a queuing system at the SDAP sublayer. Meticulously managing the queuing system at UPF will not successfully achieve envisioned requirements if the bufferbloat happens at lower stack entities.
- Based on the queuing theory results [25] and 5G specificities, (e)5G-BDP has been designed and presented with very promising results. The utilization rate is maintained close to its maximum, while reducing the delay to nearly its minimum possible value. (e)5G-BDP successfully fulfills the challenges of maintaining the buffers at the optimum size, enabling a rapid packet delivery and providing high utilization rate. Additionally, we also presented the USP algorithm that surpasses BBR and CoDel, and that can be combined with any RLC queue size limiting algorithm.
- A general bufferbloat fighting mechanism such as BBR may not cover all the scenarios efficiently (e.g. 5G), as demonstrated in this Chapter 3. Hence, it may not be the most suitable solution even in environments, where the congestion control algorithm of the sender can be chosen (e.g., MEC environments).

- Solutions that obtain more information about the cellular network (i.e., (e)5G-BDP, DRQL and DynRLC) naturally surpass solutions that lack such information (i.e., CoDel, BBR and Vanilla) at the cost of adding some complexity into the system.
- A trade-off between the utilized bandwidth and the sojourn time is continuously present when addressing the bufferbloat phenomenon. It is relatively easy to use all the available bandwidth and bloat the buffers (i.e., Vanilla), or achieve low-latency and squander bandwidth, while achieving both is a non-trivial problem due to the amount of factors involved. In essence, the pacing rate should work as close as possible to the  $\beta$  point from Fig. 3.3.

## CHAPTER 4

---

# Addressing the latency generated by the RLC sublayer segmentation/reassembly procedure

---

*In this chapter we, first describe the RLC sublayer according to 3GPP, and we underline the enhancements that occurred in 5G. Following, we analyze the segmentation/reassembly procedure, and we present its effect on the latency. We then propose an elastic scheduling algorithm for minimizing the RLC segmentation/reassembly procedure latency, while still respecting the fairness between different entities. The evaluation framework is later described, and this chapter ends with the evaluation of the proposed algorithm (i.e., EQP) along with the FP scheduler in a testbed to verify its suitability and effectiveness under realistic conditions of use by considering MCS variations, slices, different traffic patterns and off-the-shelf equipment.*

### **Contributions:**

[J2] M. Irazabal, E. Lopez-Aguilera, I. Demirkol, R. Schmidt and N. Nikaiein, "Preventing RLC

Buffer Sojourn Times in 5G", IEEE Access, March 2021. (Area: Telecommunications; Quartile Q1; IF: 3.745).

## 4.1 RLC Sublayer

RLC is the sublayer where the last buffer before the transmission in downlink procedure rests, and is configured by the RRC. RLC sublayer is above the MAC and below the PDCP sublayer as seen in Fig. 2.2, and it can be configured in 3 modes: Transparent Mode (TM), UM and AM. TM is the simplest mode in which a RLC entity can be configured. Its only component is the transmission buffer as observed in Fig. 4.1. TM only maps to the Broadcast Control Channel (BCCH), the Paging Control Channel (PCCH) and the Common Control Channel (CCCH) logical channels, which are used to transmit control information, in contrast to UM and AM modes that transmit data information.

As seen in Fig. 4.2, the UM has different phases. In the first phase, a RLC header is generated and the SDU received from the PDCP is stored in a queue. Once notified by the MAC sublayer, the RLC UM entity shall segment the RLC SDU if needed, form the PDU with the appropriate headers and forward it to the MAC sublayer. When receiving the PDU, the RLC UM shall remove the RLC header, detect if a loss of the RLC SDU at lower layers occurred, and reassemble the previously segmented SDU. The reassembly is contingent and only happens if the segmentation previously occurred. The RLC UM entity can only be configured to receive/submit through the Dedicated Traffic Channel (DTCH) logical channel.

The most feature complete mode in which a RLC entity can be instantiated is the AM. As seen in Fig. 4.3, PDCP PDUs are received, stored and when a transmission opportunity is notified by lower layers, the appropriate header is appended, and if needed, segmentation occurs. However, AM differs from UM in the fact that forwarded packets are copied into a retransmission buffer. When packets are received, the RLC AM entity at the receiver can detect if a packet was lost and request a retransmission when needed. On the contrary if no loss happened, the RLC header is removed and, if needed, a SDU reassembly happens. Note again that a reassembly only occurs if a segmentation previously happened. RLC AM can be configured to send packets through the DTCH and the Dedicated Control Channel (DCCH) channels. RLC AM capability is especially interesting in lossy environments since the most common TCP congestion control algorithm (i.e., TCP Cubic) interprets a packet lost as a congestion symptom, and consequently



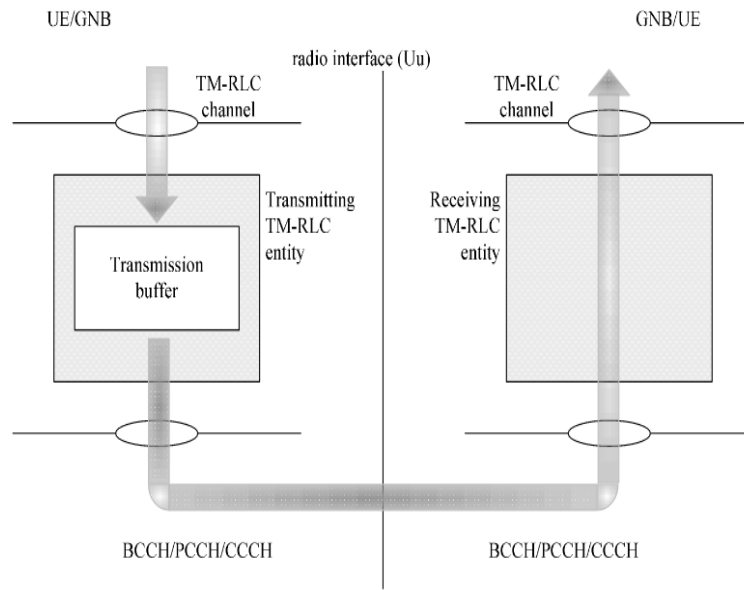


Figure 4.1: RLC TM from 3GPP 38.322 [4].

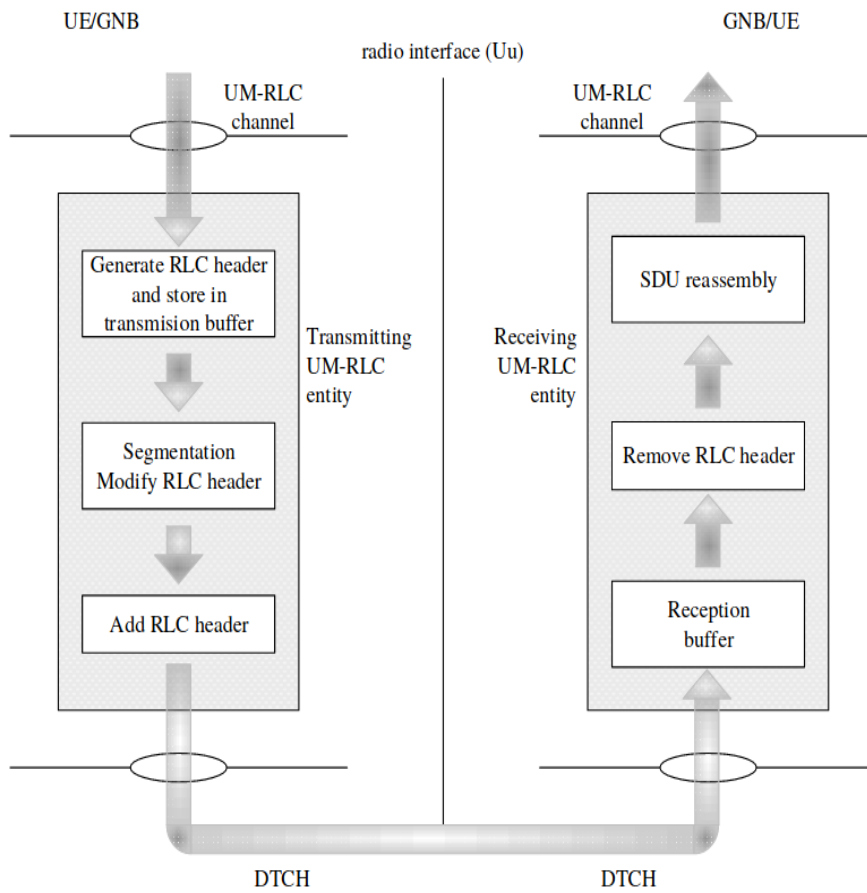


Figure 4.2: RLC UM from 3GPP 38.322 [4].

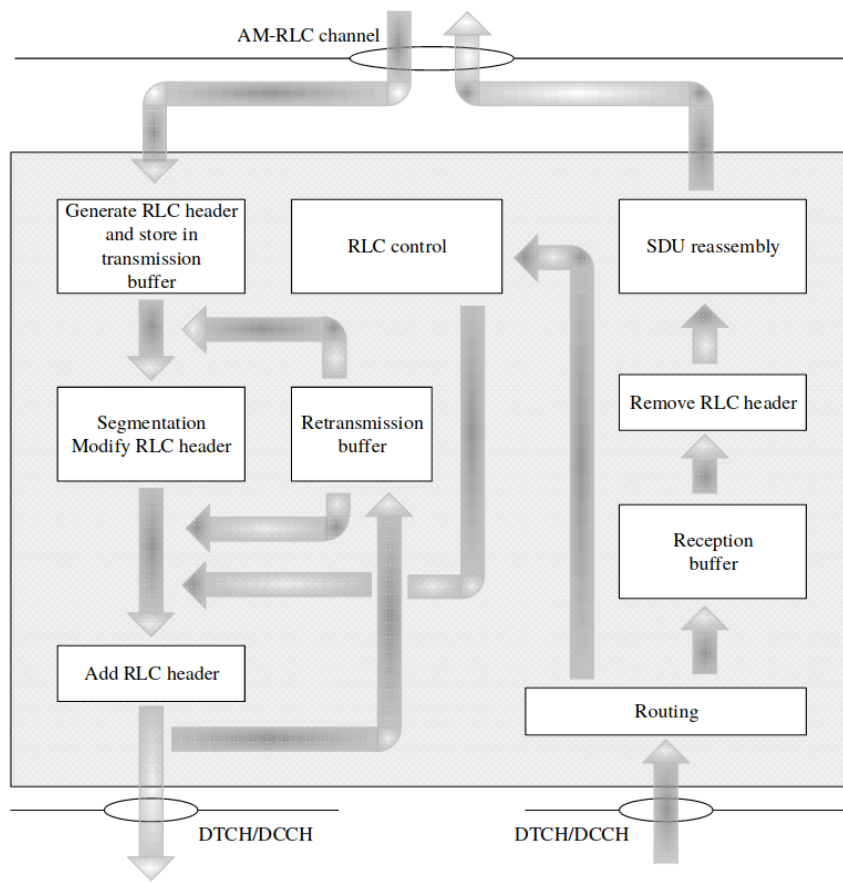


Figure 4.3: RLC AM from 3GPP 38.322 [4].

reduces its rate. This scenario can lead to underutilization of the channel capacity as the packet lost did not occur due to a buffer overflow, and thus, a saturation sign, but rather by a wireless transmission error.

## 4.2 RLC Stack Changes in 5G

3GPP has also introduced stack network improvements at the RLC sublayer in 5G [4] in comparison with LTE [5] aiming to decrease the latency. At LTE, the RLC PDU header for data transmission consists of a fixed (i.e., one or two bytes depending on the configuration) and an extension part. The fixed part is byte aligned and consists of a Framing Info (FI), Extension bit (E) and a Sequence Number (SN). The extension part is only present when more than one packet is assembled and is composed by an E and a Length Indicator (LI), and is also byte aligned. 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

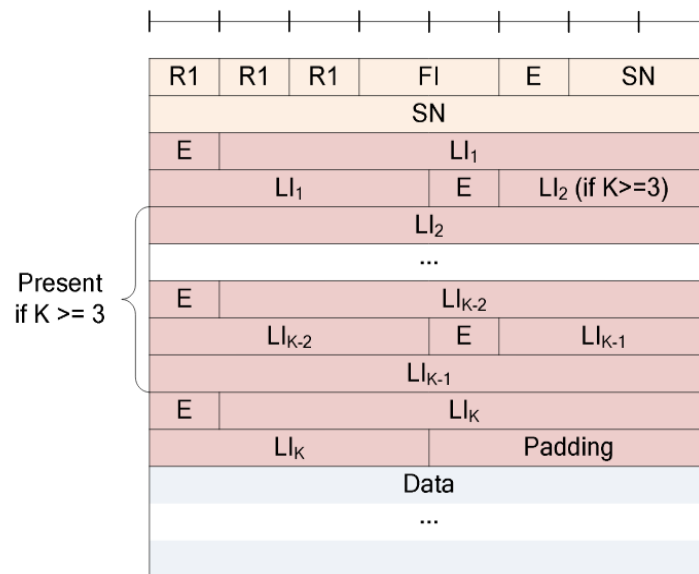


Figure 4.4: RLC UM PDU with 10 bit SN (Odd number of LIs, i.e.  $K = 1, 3, 5, \dots$ ) from 3GPP 36.322 [5].

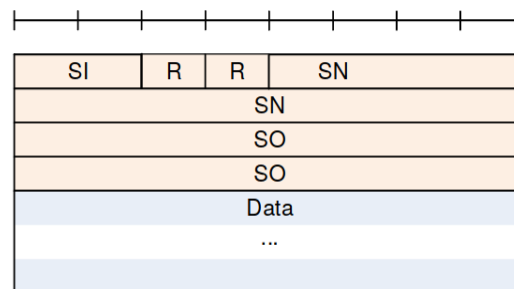
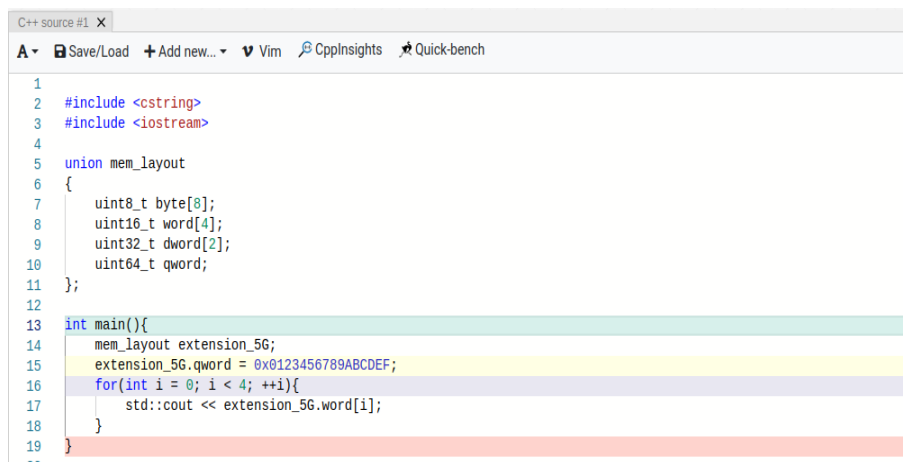


Figure 4.5: RLC UM PDU with 12 bit SN and with SO from 3GPP 38.322 [4].

(for PDUs larger than  $2^{11} = 2048$  bytes) according to the configuration. In case that a 11 bit LI is configured and the header consists of an odd number of LI(s), four padding bits follow the last LI, so that the RLC PDU is byte aligned (e.g., three E-LI pairs occupies  $3 \times (11 + 1) + 4 = 40$  bits or five bytes) as observed in Fig. 4.4. In 5G the extension part has been renamed as Segment Offset (SO) and consists of 16 bits for all cases as observed in Fig. 4.5. It indicates the absolute position of the SDU 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 SDUs starting positions is necessarily smaller than the absolute SDUs starting position.

However, in modern processors the minimum amount of data that can be accessed is one byte.

Therefore, if the information to be obtained lays between 2 bytes (e.g., with three 12 bits E-LIs pairs, the information of the second SDU 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 bytes), some bit manipulation assembly instructions need to be generated to access the value. 5G simplifies this process as the SDU starting position information is byte aligned, and thus, it can be directly accessed, sacrificing some throughput to reduce the latency. This can be observed in Fig. 4.6 and 4.7, where a possible implementation to access the extension header part in 5G is presented in C++ along with the assembler code generated. There, accessing a word (i.e., two bytes) can be done directly as shown at line 10 from Fig. 4.7.

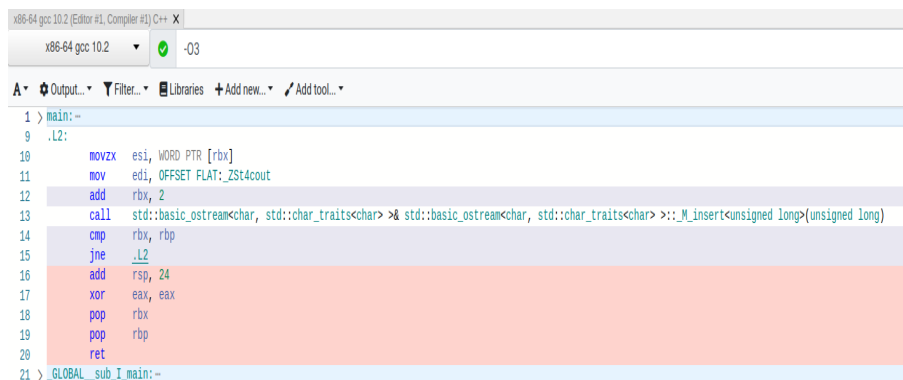


```

1
2 #include <cstring>
3 #include <iostream>
4
5 union mem_layout
6 {
7     uint8_t byte[8];
8     uint16_t word[4];
9     uint32_t dword[2];
10    uint64_t qword;
11 };
12
13 int main(){
14     mem_layout extension_5G;
15     extension_5G.qword = 0x0123456789ABCDEF;
16     for(int i = 0; i < 4; ++i){
17         std::cout << extension_5G.word[i];
18     }
19 }

```

Figure 4.6: Possible C++ code for accessing RLC's header information in 5G. 8 bytes are needed to indicate the absolute position of 4 SDUs.



```

1 > main:--
9 .L2:
10 movzx esi, WORD PTR [rbx]
11 mov edi, OFFSET FLAT: ZSt4cout
12 add rbp, 2
13 call std::basic_ostream<char, std::char_traits<char>> & std::basic_ostream<char, std::char_traits<char>>::_M_insert<unsigned long>(unsigned long)
14 cmp rbp, rbp
15 jne .L2
16 add rsp, 24
17 xor eax, eax
18 pop rbp
19 pop rbp
20 ret
21 > _GLOBAL_sub_I_main:--

```

Figure 4.7: Intel assembler code produced by GCC 10.2 with the optimization flag -O3 for 5G's RLC extended header.

In 5G implementation, 8 bytes (i.e., four words) are needed for attaching 4 SDUs. On the other hand, at Fig. 4.8 and 4.9, a possible implementation for accessing the extension header part in

LTE is shown, along with its generated assembler code. In LTE, the C++ code is considerably more complicated and so is the assembler, due to the bit arithmetic. A similar implementation to the one shown in Fig. 4.8 can be found for LTE implementations in OAI<sup>1</sup> and srsLTE<sup>2</sup>. LTE implementation only needs 6 bytes to transmit 4 SDUs (i.e.,  $12 \times 4 = 48$  bits or 6 bytes), at the cost of generating a larger and more inefficient code. However, this latency reduction in 5G implementations, may not play an important role if another phenomenon that contributes to augment the delay in 5G is ignored: the segmentation/reassembly procedure.

```

C++ source #1 X
A Save/Load + Add new... Vim CppInsights Quick-bench
1
2 #include <cstring>
3 #include <iostream>
4
5 union mem_layout
6 {
7     uint8_t byte[8];
8     uint16_t word[4];
9     uint32_t dword[2];
10    uint64_t qword;
11 };
12
13 int main(){
14     mem_layout extension_4G;
15     extension_4G.qword = 0x0123456789ABCDEF;
16     for(int i = 0; i < 4; ++i){
17         uint16_t higher = extension_4G.byte[i];
18         uint16_t lower = extension_4G.byte[i+1];
19         if(i%2 == 0){
20             higher &= 0x007F; // get rid of E bit
21             higher <<= 4; // 7 bits from higher
22             lower >>= 4; // 4 bits from lower
23         } else {
24             higher &= 0x0007; // get rid of E bit
25             higher <<= 8; // 3 bits higher and 8 bits from lower
26             ++i;
27         }
28         const uint16_t r = higher | lower;
29         std::cout << r;
30     }

```

Figure 4.8: Possible C++ code for accessing RLC's header information in LTE. 6 bytes are needed to indicate the relative position of 4 SDUs.

<sup>1</sup>File openair2/LAYER2/RLC/UM\_v9.3.0/rlc\_um\_segment.c.

<sup>2</sup>File lib/src/upper/rlc\_um.cc.

```

x86-64 gcc 10.2 [Editor #1, Compiler #1] C++ X
x86-64 gcc 10.2 -O3
A Output... Filter... Libraries + Add new... + Add tool...
1 > main:--
12 .L11:
13 and esi, 127
14 shr ax, 4
15 mov edi, OFFSET FLAT:_ZSt4cout
16 sal esi, 4
17 or esi, eax
18 movzx esi, si
19 call std::basic_ostream<char, std::char_traits<char> >& std::basic_ostream<char, std::char_traits<char> >::_M_insert<unsigned long>(unsigned long)
20 cmp ebp, 4
21 je .L3
22 mov ebx, ebp
23 .L4:
24 movsx rax, ebx
25 lea ebp, [rbx+1]
26 movzx esi, BYTE PTR [rsp+8+rax]
27 movsx rax, ebp
28 movzx eax, BYTE PTR [rsp+8+rax]
29 .L5:
30 test bl, 1
31 je .L11
32 and esi, 7
33 add ebx, 2
34 mov edi, OFFSET FLAT:_ZSt4cout
35 sal esi, 8
36 or esi, eax
37 movzx esi, si
38 call std::basic_ostream<char, std::char_traits<char> >& std::basic_ostream<char, std::char_traits<char> >::_M_insert<unsigned long>(unsigned long)
39 cmp ebx, 3
40 jle .L4
41 .L3:
42 add rsp, 24
43 xor eax, eax
44 pop rbx
45 pop rbp
46 ret
47 > _GLOBAL__sub_I_main:--

```

Figure 4.9: Intel assembler code produced by GCC 10.2 with the optimization flag -O3 for LTE's RLC extended header.

### 4.3 Segmentation/Reassembly Procedure in LTE and 5G

The segmentation/reassembly procedure is one of cellular network's specificities, where packets at RLC are segmented and latter reassembled [4], if the amount of bytes in the next RLC SDU exceeds the amount of bytes requested by the MAC. As seen in Fig. 4.2 and 4.3, 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 forwarding them to the next upper sublayer (i.e., receiver's PDCP in the downlink). 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 available RBs. The maximum size of an IP packet transmitted through Ethernet is of 1500 bytes, while a base station with 5 MHz bandwidth under best radio conditions (i.e., 28 MCS) can transmit approximately 2289 bytes every TTI of 1 ms [12]. Since 2289 is not congruent 0 modulo 1500 or  $2289 \equiv 0 \pmod{1500}$  does not hold, it can easily be conjectured that a myriad of fragmented packets are formed at the RLC sublayer, thus

augmenting the delay.

In contrast, in other protocols such as Ethernet (i.e., IEEE 802.3) or Wireless Fidelity (Wi-Fi) (i.e., IEEE 802.11), the data is mostly transmitted asynchronously. In Ethernet, the MTU size is 1500 bytes<sup>3</sup> [83], while in Wi-Fi it reaches 2304 bytes [84]. Even though modern versions of Ethernet do not employ any collision detection mechanism, Wi-Fi uses a collision avoidance mechanism to orchestrate the access of multiple stations to the common wireless medium. An acknowledgment frame from the receiver is used to confirm that the data arrived correctly. IEEE 802.11 increases its probability of successfully transmitting the packet in a noisy channel through packet fragmentation. However, the asynchronous nature of accessing the channel, compels to aggregate frames if full bandwidth is to be achieved [46], and therefore, the fragmentation procedure is rare in comparison with the RLC segmentation/reassembly mechanism that arises mostly due to the cellular network synchronous nature. Therefore, the segmentation/reassembly procedure can be mostly considered as a cellular network specificity.

#### 4.4 Avoiding the Segmentation/Reassembly Procedure: EQP

According to 3GPP [4], and as thoroughly explained in the previous Section 4.3, packets at the RLC sublayer will be segmented if the RLC SDU size is larger than the bytes requested by the MAC sublayer. When a packet is segmented, a fraction of the packet is transmitted to the receiver, while the remainder rests at the sender's RLC buffer. The received segments are retained at the receiver's RLC buffer until the rest of the packet's segments arrive, a reassembly occurs and the RLC SDU is forwarded to the PDCP sublayer. The information is not forwarded to the PDCP sublayer until the packet is completely formed, which increases the delay. Hence, a strict spectrum based distribution (i.e., fixed amount of RBs per RLC buffer every TTI) leads to a plethora of packets waiting at the receivers' RLC buffers, which significantly augments the delay. In Fig. 4.10, the problem is depicted with 4 RLC buffers, each containing one RLC SDU packet of size 4 RBs, that appears in a static MCS scenario, where the SDU size in bytes is converted to RBs, and the bandwidth is equal to 4 RBs/TTI. In the Fixed Partition (FP) approach, one fourth of each RLC buffer packet is sent every TTI. This is the baseline algorithm used in a

---

<sup>3</sup>The standard explicitly talks about octets rather than bytes. Through this thesis, no differentiation is made and the bytes are considered as 8 bits objects or octets.

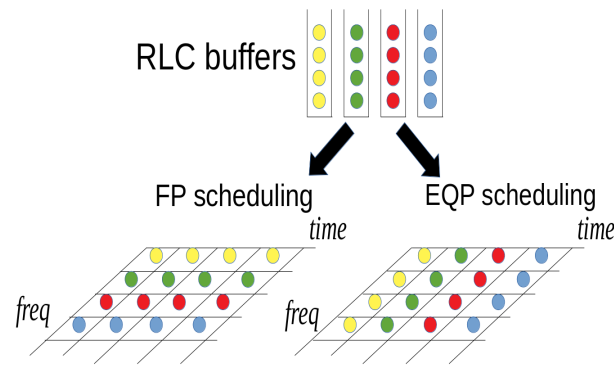


Figure 4.10: Illustration of FP and EQP RB distribution algorithms.

frequency slicing scenario. For example, one of the most advanced tools for research in slicing known as FlexRAN [85], can create different slices, assign a percentage of the total amount of RBs, and attach different UEs to different slices with diverse requirements. FlexRAN jointly with OAI uses a FP approach, for the three downlink schedulers that it supports (i.e., proportional fair, round robin and maximum throughput<sup>4</sup>), where RBs are assigned beforehand to different slices that may be configured with diverse schedulers.

This leads to segmented packets that cannot be completely forwarded until the 4<sup>th</sup> TTI, leading to an average delay of  $(4 \times 4 \text{ TTIs} / 4 \text{ packets} = 4 \text{ TTI/packet})$ . Contrarily, a segmentation/re-assembly 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  $((1 + 2 + 3 + 4) \text{ TTI} / 4 \text{ packets} = 2.5 \text{ TTI/packet})$ . In essence, it is faster to assign all the RBs to a queue, so that a packet can continue its way in the stack, rather than distributing the RBs evenly among all the packets and retain the information at the RLC sublayer. We amend to address this segmentation/reassembly delay problem and propose the EQP algorithm.

RLC SDUs wait at the RLC sublayer until a MAC notification with the requested bytes arrives. The RLC buffer is a FIFO queue [17] [18] since packets within a DRB should be treated equally [1], and, therefore the MAC scheduler can only access the most recently inserted packets after it has pulled the older ones. This introduces a precedence constraint of the packets that belong to the same queue (i.e., packets from the same queue can only be egressed following the arrival

<sup>4</sup>File openair2/LAYER2/MAC/pre\_processor.c



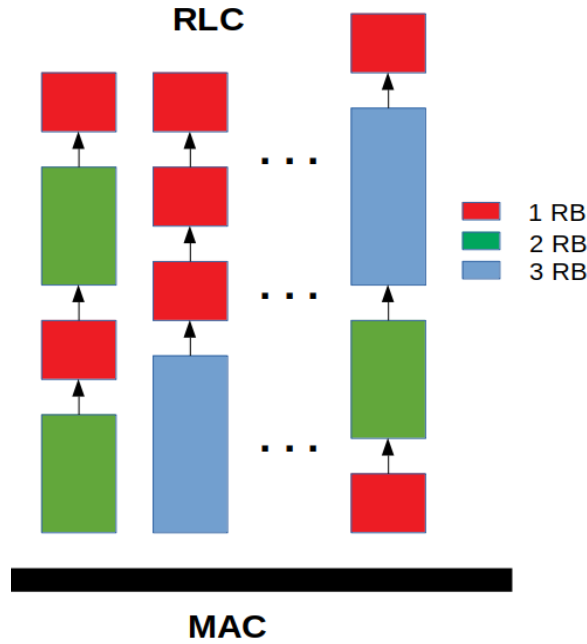


Figure 4.11: RLC DRB buffers with different sized packets. The red, green and blue colors represent 1, 2 and 3 RB sized packets, respectively. The arrow indicates the precedence constraint between packets.

order). The result is a partially ordered set or *poset* as illustrated in Fig. 4.11.

Consider the partially ordered set  $(X, \leq)$  composed by  $N$  packets at the RLC sublayer  $x_1, x_2, \dots, x_N$ , where  $N \in \mathbb{N}$ . Since packets are egressed in a FIFO order from every RLC buffer, a precedence constraint exists (i.e., to egress a particular packet all the packets that arrived before it have to be dequeued), and therefore, the precedence relation  $(i, j)$  exists in  $X$  if item  $i$  can only be pulled once  $j$  has been pulled as the precedence constraint arrow shows in Fig. 4.11. The capacity of the RAN in the current TTI is represented with a  $C \in \mathbb{N}$ , and the set  $R$  composed of  $r_1, r_2, \dots, r_N$  contains the ceil size in RBs of the packets. As an example, and without loss of generality, let us suppose that a RB can transport 88 bytes (i.e., the approximate number of bytes per RB with 28 MCS [12]), 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 of  $23 - 18 = 5$  RBs instead of  $\lceil 500/88 \rceil = 6$  RBs. Therefore, that queue would contain a 18 RB packet followed by a 5 RB packet. The objective function to intelligently avoid the segmentation/reassembly procedure given the 5G model is to maximize

the number of packets pulled from the RLC buffers that belong to a slice, given a capacity  $C$  during a TTI, which mathematically can be expressed as:

$$\max \left( \sum_{i=1}^N x_i \right) \quad (4.1)$$

s.t.

$$\sum_{i=1}^N x_i r_i \leq C \quad (4.2)$$

$$x_i \in \{0, 1\}, \forall i \in N \quad (4.3)$$

$$x_i \leq x_j, \forall (i, j) \in X \quad (4.4)$$

The objective function is given in (4.1), where the number of packets transmitted is to be maximized. Thus,  $x_i$  can be either 0 or 1 according to constraint (4.3), where a 1 represents that the packet is to be submitted and a 0 represents that the packet will not be submitted. Constraint (4.2.) models the requirement that the sum of the RBs, where  $r$  represents the amount of RBs of a packet, of the selected packets cannot surpass the capacity  $C$ . Lastly, (4.4.) models the precedence constraint, where if a precedence constraint exists between two packets  $(i, j)$  (i.e., they belong to the same queue and the  $x_j$  packet arrived before the  $x_i$  packet), 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 and studied problem: the *precedence constrained knapsack problem* (PCKP)<sup>5</sup>, which is also known as the open pit mining problem. Even though the PCKP is known to be a  $\mathcal{NP}$ -complete [87] problem, 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 (i.e., in 5G the maximum number of RBs is limited to 275 [72]). This fact mitigates the  $C$  variable's effect in the scheduling algorithm, as no more than the equivalent to 275 RBs per queue has to be considered, and Ethernet packets will be mostly composed by packets larger than one RB<sup>6</sup>. However,  $N$  depends on the number of connected UEs, which in

<sup>5</sup>The Weighted Completion Time and Chains [86] algorithm, which aims to optimally schedule different jobs with precedence constraints, does not achieve the optimum scheduling 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).

<sup>6</sup>Since the  $RB$  set contains the ceil value in RBs of the packets, adding a new small packet could result in a 0 size

commercial base stations can be of the order of  $10^4$  [88], which should be multiplied by the maximum number of DRBs (i.e., 30 [16]) per UE.

The objective function (4.1.) applied to the RB scheduler, maximizes the number of unsegmented packets satisfying the precedence constraint and, therefore, the delay suffered by segmentation is minimized. This objective function favors small packets over large ones. However, small, in this context is translated to the number of RBs needed, which does not map directly to the number of bytes of the packets as the channel radio quality has to be considered. 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. Fortunately, the problem of the fairness in a packet based network has been widely studied during the last decades [48] [89]. In [48], the DRR 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 summed 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 possible deficit or past unfairness is fixed in future egress opportunities, achieving fairness. However, in a typical spectrum sharing scenario, the percentage of the total RBs is agreed among 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) [1]. 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.

To bring fairness to our algorithm, we introduced a variable named quantum  $q$ . We imple-

---

RB packet. For example, if a packet contains 100 bytes and a RB can transport up to 89 bytes, the number of RBs to transmit it are  $\lceil 100/89 \rceil = 2$ . If the next packet's size is 50 bytes, the number of RBs to transmit both packets are  $\lceil (100 + 50)/89 \rceil = 2$ . Thus, adding a second packet does create a 0 RBs 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, and thus, the situation described above is uncommon.

mented it per slice, even though it can be generalized to other resource scheduling scenarios. We applied a very similar semantic to FlexRAN's [85] to the concept of slices, with the additional functionality, that slices should be considered per DRB rather than per UE. The value of  $q$  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 for a slice). Our algorithm let 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 concrete TTI to avoid packet segmentation and, thus, lessen the delay. However, to avoid unfairness, a limit to the amount of quantum is applied. Such 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 limit, the access to new RBs is denied. The sum of the resource quantum 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, if low-latency packets arrive, the slice lacks the capability of borrowing enough resources to transmit them rapidly. In the last years, different buffer management policies for packet switches [90] using the competitive analysis [91] have flourished. There, the online performance against the optimal offline performance for any input packet sequence is analyzed. Even though the cellular network segmentation/reassembly problem cannot be reduced to a known scheduling buffer model due to its complexity, to the best of the author's knowledge (e.g., different packet sizes or dynamic radio link capacity), some results can be applied. At [92], 5 different buffer management policies are presented for packets that can have two different values (i.e., high or low). The most successful policy (i.e., Dynamic Flexible Partition) accepts low value packets according to the amount of low value packets enqueued and it's amount of free slots through an exponential function. The reasoning being as follows. Since in an online algorithm, the future packet sequence is unknown, assign the resources to packets that do not contribute to the total reward significantly cautiously, as the algorithm may need the resources in the near future for more profitable packets. We use the same approach to discourage an already indebted slice from acquiring more resources. The amount of bytes of the packets are augmented through an exponential function according to the quantum already used. In this manner, acquiring more quantum is discouraged, especially if the borrowed quan-

tum approaches the maximum, and only considered if significant larger amount of packets are dequeued. With this mechanism, the greedy effect of selecting the packets from the slice with smaller size packets is limited, thus achieving the objective of reducing the packet segmentation effect, while maintaining the fairness in the SLA within a small target deviation. To achieve such objectives, we propose the EQP mechanisms detailed in Algorithm 8.

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. Note that the amount of RBs that a packet consist on, will be dynamically adjusted according to the radio channel conditions. 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 4, Algorithm 8 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 is selected (i.e., the Radio Network Temporary Identifier (RNTI) is a temporal identifier for a UE). 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 maximum quantum limit, the RBs are distributed considering 5G specificities. In 5G as well as in LTE, the RBs cannot be scheduled individually, but rather in Resource Block Groups (RBG) [82]. The objective for avoiding indebted above 75% of the maximum quantum limit 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 at line 20, 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. Even though such scheme may not appear optimal due to the fact that RBs are left empty even though one of the DRBs contains data, the SLA may also be subject to monetizing, and thus, a slice will not acquire all the available bandwidth unless so agreed with the infrastructure provider.

---

**Algorithm 8** *Elastic Quantum Partition (EQP)*.

It maps the free RBs to non empty RLC buffers.

---

```

1: procedure EQP(total_rbs, active_rntis)
2:   assign_quantum_slices();
3:   idx_q = generate_rbs(total_rbs, active_rntis);
4:   out_arr = seg_avoid(idx_q, 0, total_rbs, out_arr); // Alg. 9
5:   for all pkt, rnti, slice ∈ out_arr do
6:     assign_rbs(rnti, pkt)
7:     reduce_slice_quantum(slice, pkt)
8:     total_rbs − = pkt
9:   end for
10:  sort_slices()
11:  rnti = next_rnti()
12:  while total_rbs > 0 ∧ slice_quantum(rnti) > −0.75 × limit_quantum do
13:    assign_rbs(rnti, min_rbg)
14:    reduce_slice_quantum(slice, min_rbg)
15:    total_rbs − = min_rbg
16:    sort_slices()
17:    rnti = next_rnti()
18:  end while
19:  if total_rbs > 0 then
20:    map_last_RB_slices()
21:  end if
22: end procedure

```

---

**Algorithm 9** EQP *seg\_avoid*.

It selects a permutation of packets according to their size and the slice quantum that does not cause segmentation.

---

```

1: function SEG_AVOID(idx_q, idx_pkt, capacity, out_arr)
2:   if capacity == 0 then
3:     return out_arr
4:   end if
5:   if idx_q > max_idx_q then
6:     return out_arr
7:   end if
8:   if is_memoized(idx_q, idx_pkt, capacity) then
9:     return memoized(idx_q, idx_pkt, capacity)
10:  end if
11:  p_size = pkt_size(idx_q, idx_pkt)
12:  t_capacity = adjust_rbgs(capacity, out_arr, p_size)
13:  if t_capacity < 0 then
14:    idx_q = next_idx_q(idx_q)
15:    idx_pkt = 0
16:    return seg_avoidance(idx_q, idx_pkt, capacity, out_arr)
17:  end if
18:  idx_pkt+ = 1
19:  if idx_pkt == max_idx_pkt(idx_q) then
20:    idx_q = next_idx_q(idx_q)
21:    idx_pkt = 0
22:  end if
23:  act_pkt = {idx_q, idx_pkt}
24:  take = seg_avoid(idx_q, idx_pkt, t_capacity, out_arr + act_pkt) /* select the pkt */
25:  dont_take = seg_avoid(next_idx_q(idx_q), 0, capacity, out_arr) /* try next pkt*/
26:  out_arr = dont_take
27:  if len(take) > len(dont_take) then
28:    out_arr = take
29:  else if len(take) == len(dont_take) ∧ quantum(take) > quantum(dont_take) then
30:    out_arr = take
31:  end if
32:  memoization(idx_q, idx_pkt, capacity, out_arr)
33:  return out_arr
34: end function

```

---

Algorithm 9 presents the segmentation avoidance algorithm. It consists in a recursive algorithm where a memoization<sup>7</sup> (i.e., at lines 8, 9 and 32) 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.

Due to the recursive nature of Algorithm 9, we will start presenting it from the middle rather than from the beginning. The core of the segmentation avoidance algorithm resides at lines 24 and 25. 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 24), or ignore it and jump to the next queue maintaining the available capacity (i.e., *dont\_take*) (line 25)). At line 27, 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 29 (i.e., every packet belongs to a slice with a quantum associated, so the larger sum of the corresponding quantum indicates the permutation of packets that reside at slices that lend more RBs). Such measurement balances the quantum values of the slices and achieves more 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 32 before returning.

As input Algorithm 9 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 9 are coded in the first lines (i.e., lines 1 – 7). At line 2, 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 5, the end of the queues (i.e., no more queues to consider) are checked, and if the end is reached, again, the current packets' permutation (i.e.,  $out\_arr$ ) is returned. At line 8, whether the packet permutation has already been resolved is checked, thus avoiding unnecessary work and reducing the algorithm complexity. At line 11, the packet size in RBs according to its queue index (i.e.,  $idx\_q$ ) and packet

---

<sup>7</sup>Memoization is not a typo from the word *memorization* but rather an optimization technique



index (i.e.,  $idx\_pkt$ ) is acquired (i.e.,  $p\_size$ ). At line 12, 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 16, after updating the queue and packet index. If the current packet is the last one in the queue, the queue and the packet indexes are updated (i.e.,  $idx\_q$  and  $idx\_pkt$  at lines 20 and 21).

In summary, Algorithm 9 generates a packet permutation where the number of packets to forward without generating segmentation is maximized (i.e., line 27). In case that two permutations would forward the same number of packets, Algorithm 9 selects the permutation with the largest quantum (i.e., line 29), and thus, it reduces the unfairness between slices, as the slices with larger quantum lent more RBs in the past. Lastly, Algorithm 9 is aware of the RBG distribution and adjusts the remaining capacity accordingly through the function *adjust\_rbg*s.

## 4.5 Evaluation Framework and Experimental Results

Following, the evaluation framework is described and the obtained results are presented.

### 4.5.1 Evaluation Framework

To evaluate EQP, the OAI testbed shown in Fig. 3.18 has been used instead of the emulator due to the larger 5G feature subset of the former. The testbed is composed of 2 COTS UEs, a B-200 Ettus USRP, and a PC running OAI, with our proposed algorithms. The proposed EQP solution has been compared against the default resource distribution in OAI (i.e., FP) for slicing, where the SLA distributes the spectrum according to a percentage of the total RBs without considering the packet sizes. Independently analyzing the delay aggregated by the segmentation/reassembly procedure would be pointless since the bufferbloat effect is order of magnitudes larger. In fact, it can add seconds of delay as show in Fig. 3.19. Therefore, the e5G-BDP algorithm has been chosen as the bufferbloat avoidance algorithm due to its superior results when compared with its direct competitors (i.e., Chapter 3). This challenges EQP as the measured outcomes are the result of a bufferbloat avoidance algorithm in conjunction with a segmentation/reassembly algorithm. However, we are exposing EQP to a more realistic scenario, and therefore, the outcomes are also more rewarding. Two UEs are assigned to two different slices, where the

SLA is provisioned with 75%, 50% and 25% of the total floor number of RBs (e.g., for 25 RBs this is translated to  $\lfloor (25 \times 0.75) \rfloor = 18$ ,  $\lfloor (25 \times 0.50) \rfloor = 12$  and  $\lfloor (25 \times 0.25) \rfloor = 6$  RBs per TTI, respectively.).

The amount of different services that simultaneously run on contemporary UEs create different flows that share buffers along the stack. Therefore, we evaluate 1, 2, 4 and 8 VoIP flows in parallel in a 25% RB slice (i.e., with 6 RBs) along a bulky flow, while the other slice is fed with a bulky traffic flow and uses 75% of the RBs (i.e., 18 RBs).

We also test the effect of the quantum through five different maximum 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. We set the function that increases the size of the packets exponentially to  $5 \times fraction^5$  to discourage considering packets from indebted slices, and a quantum value of 100 RBs was set as the default value, as no saturation was detected in any slice for the traffic patterns tested.

The dynamic throughput on the cellular networks considerably challenges any solution. Hence, the radio channel link dynamicity is tested through two 50% RB slices, where one of the UE's radio link channel is modeled according to the CQI trace from a real LTE pedestrian UE [26], and the second UE's radio link channel is modeled according to the CQI trace from a real LTE train UE [26]. OAI's default slice SLA is 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) and control RBs.

Regarding the hardware or the tools used for emulating the traffic, an analogous configuration to the one described at Section 3.6.3 has been utilized, where the low-latency flows are generated through the *irtt* tool, modeling a G.711 VoIP flow that lasts for 60 seconds, while *iperf3* creates the bulky traffic flow that models a bandwidth driven service (e.g., a software update).

#### 4.5.2 Experimental Results

Following the experimental results obtained for the EQP in different scenarios are shown.

##### Static MCS Scenario

As thoroughly analyzed in Section 4.4, RLC sublayer's capacity to segment the packets generates an unnecessary, yet avoidable delay. We evaluate the EQP algorithm against the FP RB distribu-

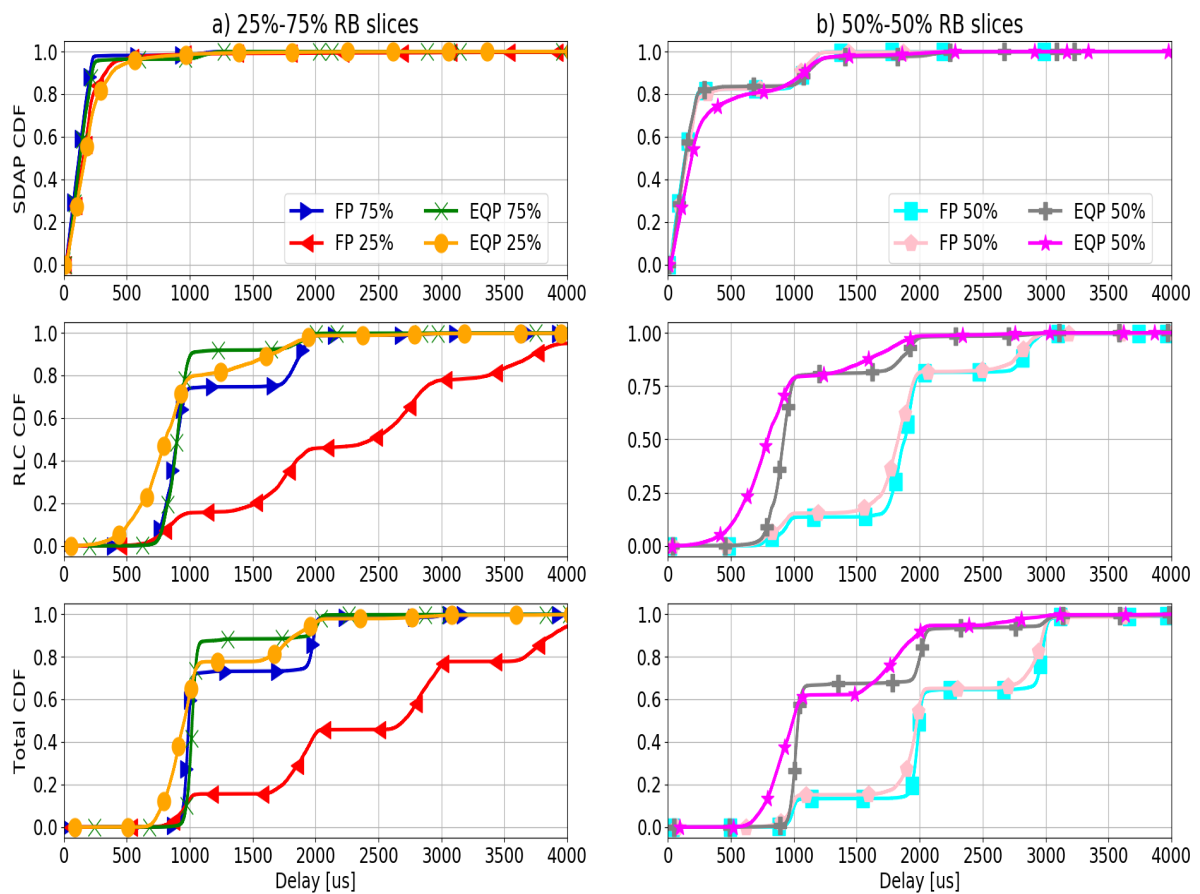


Figure 4.12: CDF of the queuing delay for VoIP packets at SDAP, RLC and the sum of both for FP and EQP in two slicing scenarios (i.e., a) 75%-25% and b) 50%-50% resource distribution) with 28 MCS .

tion algorithm in a slicing scenario with e5G-BDP algorithm as our base bufferbloat avoidance method. We first create two different slicing scenarios, one where both slices are assigned the 50% of the available RBs, and another one where the RBs are shared with a 75%-25% ratio. Since we are using a 5 MHz bandwidth or 25 RBs, the floor value of 25%, 50% and 75% is even (i.e.,  $\lfloor 25 \times 0.25 \rfloor = 6$  RBs,  $\lfloor 25 \times 0.50 \rfloor = 12$  RBs and  $\lfloor 25 \times 0.75 \rfloor = 18$  RBs), which are multiples of the RBG for this bandwidth (i.e., 2 RBs except for the last RB for a 5 MHz eNodeB). For both scenarios, a low-latency traffic flow along with a bulky traffic flow is generated as in Section 3.6.3 and the MCS index is set to 28.

As it can be observed in Fig. 4.12, 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

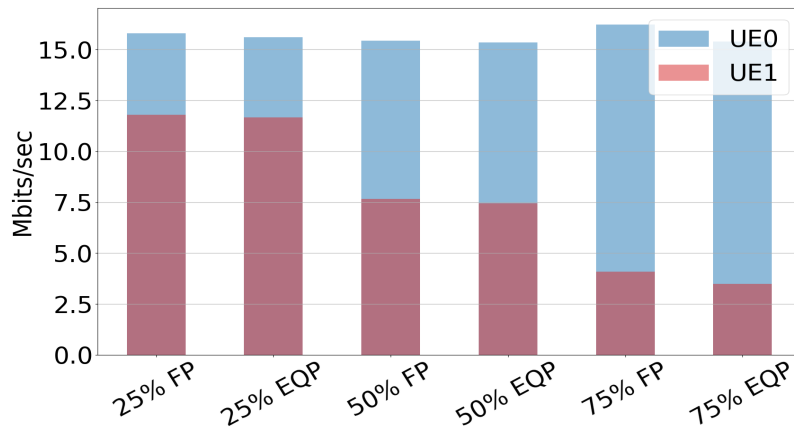


Figure 4.13: Average throughput reported by *iperf3* in two slicing scenarios (i.e., 75%-25% and 50% - 50% resource distribution) for both UEs with 28 MCS for FP and EQP.

delivered during the first 1000  $\mu s$  in contrast with the approximately 15% when FP is used. The improvement is reduced, albeit still significant, when the RBs are equally shared, with an approximate 64% vs. 14% of packets being forwarded within the first 1000  $\mu s$ , and non-negligible for the slice with 75% of the RBs, where a 88% vs. a 73% is observed.

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 at Fig. 4.12, 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 [12], so if the queue contains 1650 bytes, 42 padding bytes are added into the last RB, and therefore, 42 bytes do not transport information, thus reducing the throughput. This effect is more pronounced due to the RB distribution based in RBGs [71] (e.g., for a 5 MHz bandwidth cell, the minimum RBG size is 2). Such behavior challenges 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. 4.13, does not substantially reduce the throughput, when compared with FP for this configuration. Note that this effect could be increased if another cell bandwidth with larger minimum RBG size is used. Different packet sizes for current scenario is not expected to substantially affect the results as padding only happens if there are no more packets in the queue where the RBs are assigned, and the SDAP sublayer, is expected to have a sufficiently full buffer.

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

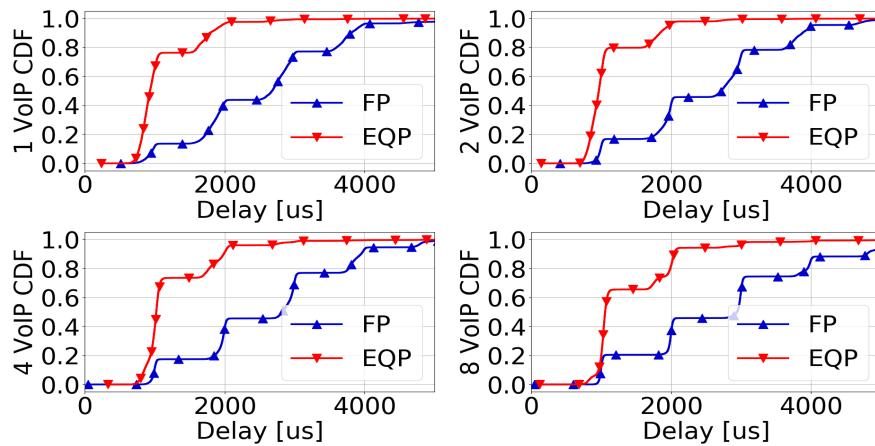


Figure 4.14: 8 VoIP flows scalability for 25% RBs using e5G-BDP.

75%-25% resource distribution, for FP and EQP is reported. OAI's default resource distribution (i.e., FP) discards the last RB (i.e.,  $[0.75 \times 25] = 18$  RBs and  $[0.25 \times 25] = 6$  RBs for the total of 25 RBs), and therefore, the average throughput of Fig. 4.13 is slightly smaller than the average throughput of e5G-BDP at Fig. 3.23.

#### Scalability Evaluations with 1, 2, 4 and 8 VoIP low-latency flows

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 25% slice contains the low-latency flows along a bulky traffic flow, while the 75% slice, only transports one bulky traffic flow. We can think of this scenario as a private network where 6 RBs were agreed between the infrastructure provider and the tenant.

As it can be observed in Fig. 4.14, 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 on average created (i.e., a packet is created every 20 ms per flow to emulate a VoIP flow [74] / 8 flows = 2.5 ms). When the interarrival packet average time approaches one TTI, 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  $\mu$ s with EQP are 98%, 98%, 96% and 94% for 1, 2, 4, and 8 VoIP flows, respectively, while for similar results (i.e., 96%, 95%, 95% and 88%) 4000  $\mu$ s are needed for FP.

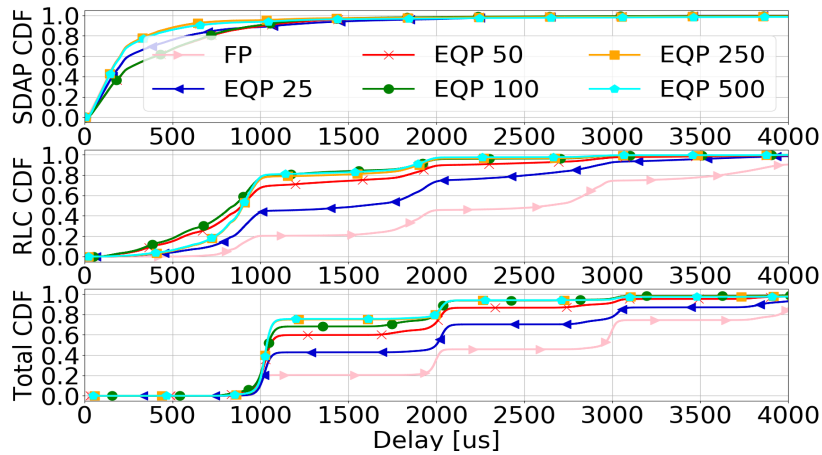


Figure 4.15: CDF of the queuing delay at SDAP, RLC, and the sum of both for VoIP packets with 8 flows, in a 25% slice with 28 MCS, for FP and EQP with 25, 50, 100, 250 and 500 RBs quantum limit.

### Effect of Quantum Limit

As explained before, 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 this end we generated two slices with a 25%-75% RB distribution and different quantum limit values. 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. 4.15, augmenting the quantum limit from 25 to 100 improves the latency of the VoIP packets. However, beyond 100 (i.e., 250 and 500) increasing the quantum limit value does not provide any latency reduction. This fact shows that it exists a boundary (i.e., 100) to the achievable latency reduction for the packet sequence tested, and thus, augmenting the quantum limit value beyond it does not report any latency 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. Even though a very interesting problem for contemporary research topics (e.g., machine learning), predicting the future traffic flows lies out of the scope of this thesis, and therefore is not analyzed in the present work.

In Fig. 4.16 the boxplot for the quantum distribution can be observed. The zero value indicates

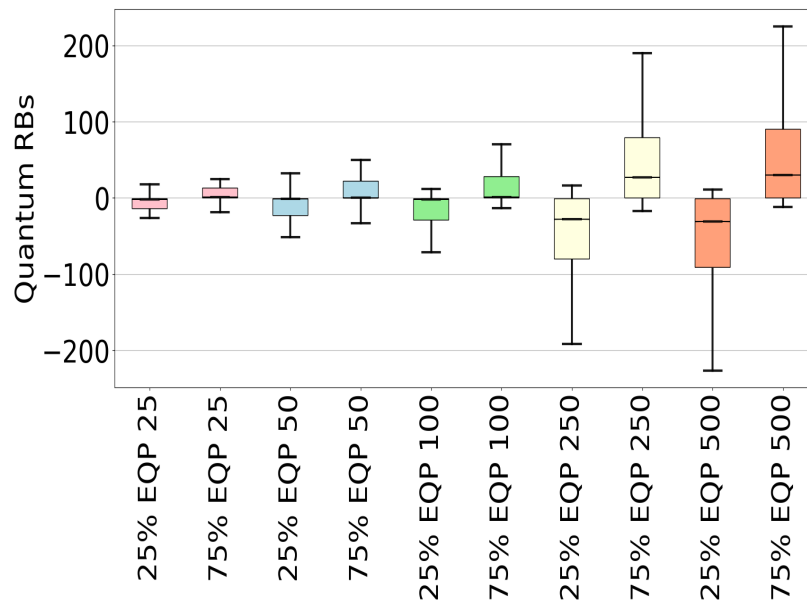


Figure 4.16: Quantum boxplot in a 25%-75% slicing scenario with 8 VoIP flows, a bulky flow in the 25% slice, and a bulky flow in the 75% slice.

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, maximize the number of forwarded packets, while the 75% slice tends to lend resources as only bulky packets traverse it, as it can be observed at Fig. 4.16. Moreover, EQP tends to be a fair RB distribution (i.e., the median slightly deviates from the equilibrium point), even if large amount of quantum RBs are available. This is an important property in an environment where the future sequence of packets is not available at the moment the RB scheduling is performed, 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.

### 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 change do not affect to 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,

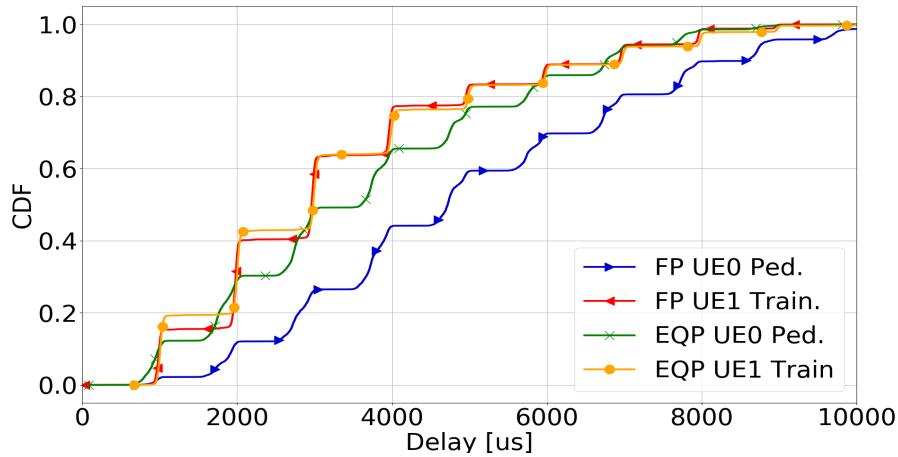


Figure 4.17: Total CDF queuing delay for VoIP packets, in two 50%-50% slices for the train and pedestrian datasets, for FP and EQP.

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 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. 4.17 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, cf. Fig. 3.25), which is strongly connected with the throughput. Therefore, and since EQP specially benefits the scenarios with scarce resources due to its borrowing-lending quantum mechanism, the difference between EQP and FP for the pedestrian slice is more explicit. Within the first 4000  $\mu s$ , EQP forwards approximately 66% and 76% of the total low-latency packets for the pedestrian and train slices, respectively, in contrast with the 44% and 77% with OAI's default approach (i.e., FP).

## 4.6 Summary

In this chapter, 5G segmentation/reassembly procedure has been presented. Its working principles have been explained, its contribution to the delay studied, a model has been exposed and an algorithm which minimizes its effect while maintaining the fairness has been presented (i.e., EQP). Following, EQP has been validated and tested with two COTS UEs belonging to two different slices in OAI, with static and dynamic MCS using data from pedestrian and train traces, thus emulating the dynamic nature of the radio link. The results show a notable improvement



in the latency when the spectrum is elastically managed through EQP.

From the segmentation/reassembly's procedure perspective and the EQP, the next conclusions can be outlined:

- In a packet based network such as the cellular network, packet sizes must be considered by the resource distribution algorithm if latency is a concern. Models ignoring such fact only generate a myriad of segmented packets at the RLC sublayer, thus increasing the total latency and impeding a rapid packet delivery.
- The idea of avoiding the segmentation/reassembly procedure is not bound to slicing. A MAC scheduler, where packets are pulled from different queues, presents the same scenario. The scheduler should wisely notify to the corresponding RLC entity (i.e., per UE and active queue) the number of RBs assigned for the next TTI.
- As in the bufferbloat problem, avoiding the segmentation may involve not utilizing all the available resources for transmission (e.g., the last RB may not be fully filled), while ignoring the segmentation/reassembly procedure produces unnecessary delay. Even though the results presented in this chapter show marginal bandwidth lost, such trade-off should always be considered.
- The quantum limit in EQP did not play a significant role in the packet sequences tested. However, if other traffic patterns are expected, increasing the quantum could be interesting at the cost of suffering larger deviations from the SLA.



## CHAPTER 5

---

### **Conclusions and Future Work**

---

*This thesis ends in this chapter. We first present a summary of the contents exposed in this document. Following, we highlight the most relevant results that emerged over the course of this thesis. Lastly, we discuss some interesting areas for future research work.*

## 5.1 Conclusions

The heterogeneity of services with different QoS in the upcoming cellular network generation will challenge the current LTE stack, as well as the protocol design to their limits. However, during this thesis we shed some light into two phenomena that are not directly related to the stack or the protocol, albeit severely contribute to the total delay experienced by data packets: the bufferbloat and the RLC segmentation/reassembly procedure.

In Chapter 2, 5G's stack QoS model has been presented. Even though considerable improvements compared with the previous cellular generation have been added, the funnel nature of 5G necessarily leads to different flows sharing buffers along the data path, following the pigeon-hole principle. Moreover, since the wireless link represents the slowest link in the data path, and it is deployed with large buffers to absorb the abrupt radio link throughput variability, the necessary conditions for the bufferbloat phenomenon to appear are satisfied. The bufferbloat can have fatal consequences for delay-sensitive services in the current cellular network stack, as demonstrated in this thesis. Therefore, a comprehensive bufferbloat survey with 5G specificities has been conducted where different techniques from distinct paradigms have been presented. It has also been shortly discussed the latency reduction of other paradigms (i.e., slicing and MEC).

The most novel bufferbloat techniques have been presented in Chapter 3. The continuous novel papers in the topic clearly indicate the lack of an optimal or satisfactory solution. In fact, as explained by [53], the impossibility of finding a non-centralized optimal solution will predictably leave the problem open for future research under different scenarios. However, some techniques have proven its validity in real network deployments. We have firstly meticulously analyzed the Linux network stack and its traffic steering tools. Out of this fruitful study, the DRQL algorithm was proposed for the cellular network with successful results. Queuing theory was later studied and applied with 5G specificities aiming to work as close as possible to the optimal pacing point [25]. The efforts of this study were consolidated in the promising (e)5G-BDP algorithm. To validate the proposed algorithms and compare them with other novel cellular network bufferbloat avoidance algorithms, an emulator from scratch was developed. Once the outcomes confirmed DRQL and (e)5G-BDP superiority, an OAI testbed was deployed and enhanced with the algorithms proposed aiming to validate the results in a cellular network with COTS UEs involved.

In Chapter 4 a complete description of 5G's RLC sublayer is conducted and its different operation modes described. Following, a delay problem that was detected due to RLC's segmentation/reassembly procedure is detailed. The data path nature of 5G inevitably segments the data packets that cannot be transmitted in the remaining RBs, which creates a packet sojourn time at the receiver's RLC sublayer, since information cannot be forwarded to upper layers until the packet is again completely formed. The fact that this problem rarely appears in IEEE 802.3 and IEEE 802.11 technologies, could explain the reason why such specificity could be kept unnoticed to the 5G research community to the best of the author's knowledge. An efficient and fair algorithm is proposed (i.e., EQP), and tested in a slicing scenario using COST UEs, OAI and a dynamic MCS index based on real traces [26]. The results unequivocally show the benefits of considering the packets' size of the data transmitted when scheduling the resources.

The major conclusions drawn from this thesis can be summarized in :

- The current 3GPP proposed 5G stack is susceptible to suffering large delays due to the bufferbloat phenomenon, which significantly contributes to the total delay suffered by the data packets.
- New capabilities need to be added into the SDAP sublayer to avoid large sojourn times of the packets. Current 3GPP SDAP sublayer specification [3] is clearly insufficient.
- There exists an optimal pacing rate where the network does not generate queuing sojourn time while serving full bandwidth [52], and unfortunately, it is not achievable through a non-centralized algorithm [53]. However, this fact should not stop us to approach this optimal pacing rate as much as possible considering 5G's specificities.
- Algorithms specifically designed for the cellular network bufferbloat (i.e., DynRLC, DRQL and (e)5G-BDP) surpass general bufferbloat avoidance algorithms (e.g., CoDel or BBR), as they are provided with extra network information that can be utilized for their benefit, enhancing 5G's QoS. Therefore, cellular network service providers should consider them when deploying new RANs, specifically (e)5G-BDP.
- The segmentation/reassembly procedure encountered at 5G's RLC sublayer inflicts an avoidable delay in the packets, when the RBs are not thoughtfully distributed. Thus, 5G specificities should be considered when distributing the RBs among the possible queues,

and segmentation and reassembly avoided as much as possible. Therefore EQP or other algorithms that consider such effects should be utilized.

- Lastly, in both of the exogenous 5G phenomena studied in this thesis (i.e., the bufferbloat and the RLC segmentation/reassembly procedure) a dichotomy between the latency and the bandwidth exists. Achieving low-latency while squandering resources or achieving full bandwidth while suffering large sojourn times is trivial. However, even though its difficulty, and as demonstrated by this thesis, working near the optimal point with marginal bandwidth reduction and queuing delay is achievable in 5G networks.

The outcomes of this thesis are promising since services with low-latency requirements can avoid two phenomena not directly connected with the 5G stack or its protocol that can still be the main cause of the delay suffered. Therefore, the solutions presented in this thesis can alleviate such problems when present.

## 5.2 Future Work

Even though the results of this thesis have not only been simulated, but emulated and integrated into a real testbed, thus validating them in a real environment, some questions and research directions remain open.

The major open aspects can be summarized as follows:

- A packet's path can contain other sources of delay that can ultimately ruin a delay sensitive service. A low-latency packet that has already suffered a large delay on the wired stack, should be prioritized in the 5G stack over other same priority low-latency packets, that did not suffer large delays in the wired stack. Hence, the study of Time Sensitive Networks (TSN) combined with 5G stack is envisioned, fulfilling the stringent requirements which new services will impose.
- A comparison with some newly deployed TCP congestion control algorithms will shed some light in the validity of addressing the latency problem through TCP's congestion control algorithm. In concrete, quantitatively measuring the advantage of using e5G-BDP over other TCP congestion control algorithms (e.g., C2TCP) deployed in MECs remains open.

- The D/D/K queue model utilized in this thesis has proven its validity. However, it has its shortcomings, and thus, a more realistic queuing model that maps more realistically to the cellular networks could be developed, aiming to achieve more accurate results.
- The channel dynamicity has been tested manipulating the MCS. However, since the real channel was better than the estimated one, some effects such as HARQ retransmissions did not appear. Therefore, a more accurate model considering such effects would certainly provide more robustness to the data here exposed, even though good results are expected, since from the e5G-BDP and EQP perspective, the only variable that would vary is the available amount of RBs, which already changes.
- The bufferbloat project community use a tool called *flent* [93] for their discussions in the IEEE 802.11 stack. Validating our tests with their tool and enhancing it when necessary, may establish a common tool for analyzing the bufferbloat effect on wireless stacks (i.e., cellular network and WiFi). Additionally more scalability tests to strenght the conclusions drawn by this thesis could be conducted.
- Current open source LTE implementations (i.e., OAI and srsLTE) traverse two times the RLC buffer to form the RLC PDU. However, it seems plausible to reduce the complexity from  $\mathcal{O}(N)$  to  $\mathcal{O}(\log N)$  augmenting the RLC buffer data structure for enabling a binary search, and thus, reduce the latency when generating the RLC PDU.
- Some cache friendly data structures (e.g., Adaptive Packed-Memory Array [94] or Cache-Oblivious Priority Queue [95]) allow efficient insertions. In concrete, in [94] an amortized element moves of  $\mathcal{O}(\log^2 N)$  is achieved, and therefore, it may well suit at the RLC buffer, enabling an efficient priority mechanism for packets with diverse QoS. In this manner, packets with different QFIs that map into a DRB could be efficiently inserted, possibly avoiding the need of segregating the traffic at upper layers (e.g., SDAP).
- The softwarization process of the future cellular networks will enable disagregating the RAN, specifically the Distributed Unit (DU) and the Central Unit (CU). Since the RLC will remain in the CU while the SDAP will remain in the DU, a delay to communicate both sublayers will be added. It remains open testing the proposed algorithms under such conditions.

- Lastly, even though some efforts have been done, merging our solutions into OAI remains open. The theory and the principles exposed in this thesis have been validated, and are reproducible. However, exposing our code in an open source project will provide the research community with a tool to enhance current understanding of the bufferbloat effect in cellular networks.



---

## Bibliography

---

- [1] 3GPP, “System architecture for the 5G System,” 3rd Generation Partnership Project (3GPP), Technical Report (TR) 23.501, Dec. 2018, version 15.4.0.
- [2] —, “NR, Overall description,” 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.300, Oct. 2018, version 15.3.1.
- [3] —, “Evolved Universal Terrestrial Radio Access (E-UTRA) and NR; Service Data Adaptation Protocol (SDAP) specification,” 3rd Generation Partnership Project (3GPP), Technical Report (TR) 37.324, Sep. 2018, version 15.1.0.
- [4] —, “NR, Radio Link Control (RLC) specification,” 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.322, Jan. 2019, version 15.4.0.
- [5] —, “Radio Link Control (RLC) specification,” 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 36.322, Jun. 2020, version 16.0.0.
- [6] T. S. Rappaport, S. Sun, R. Mayzus, H. Zhao, Y. Azar, K. Wang, G. N. Wong, J. K. Schulz, M. Samimi, and F. Gutierrez, “Millimeter Wave Mobile Communications for 5G Cellular:

- It Will Work!" *IEEE Access*, vol. 1, pp. 335–349, 2013.
- [7] C. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, Oct. 1948.
- [8] S.-Y. Chung, G. Jr, T. Richardson, and R. Urbanke, "'on the design of low-density parity-check codes within 0.0045 db of the shannon limit'," *Communications Letters, IEEE*, vol. 5, pp. 58 – 60, 03 2001.
- [9] 3GPP, "NR, Multiplexing and channel coding," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.212, Abr. 2020, version 16.1.0.
- [10] —, "Study on physical layer enhancements for NR ultra-reliable and low latency case (URLLC)," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.824, Mar. 2019, version 15.5.1.
- [11] G. Berardinelli, N. Huda Mahmood, R. Abreu, T. Jacobsen, K. Pedersen, I. Z. Kovács, and P. Mogensen, "Reliability Analysis of Uplink Grant-Free Transmission Over Shared Resources," *IEEE Access*, vol. 6, pp. 23 602–23 611, 2018.
- [12] OpenAirInterface Feature Set. [Online]. Available: [https://gitlab.eurecom.fr/oai/openairinterface5g/blob/develop/doc/FEATURE\\_SET.md](https://gitlab.eurecom.fr/oai/openairinterface5g/blob/develop/doc/FEATURE_SET.md)
- [13] D. Xu, A. Zhou, X. Zhang, G. Wang, X. Liu, C. An, Y. Shi, L. Liu, and H. Ma, "Understanding operational 5g: A first measurement study on its coverage, performance and energy consumption," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 479–494. [Online]. Available: <https://doi.org/10.1145/3387514.3405882>
- [14] S. Ha, I. Rhee, and L. Xu, "Cubic: A new tcp-friendly high-speed tcp variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, p. 64–74, Jul. 2008.
- [15] The bufferbloat project. [Online]. Available: <https://www.bufferbloat.net/projects/>
- [16] 3GPP, "NR, Radio Resource Control (RRC) protocol specification," 3rd Generation Part-

- nership Project (3GPP), Technical Specification (TS) 38.331, Apr. 2019, version 15.5.1.
- [17] OpenAirInterface. [Online]. Available: <https://www.openairinterface.org/>
- [18] srsLTE. [Online]. Available: <https://github.com/srsLTE/srsLTE>
- [19] 3GPP, "NR, Medium Access Control (MAC) protocol specification," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.321, Oct. 2020, version 16.2.1.
- [20] P. Caballero, A. Banchs, G. de Veciana, and X. Costa-Pérez, "Multi-tenant radio access network slicing: Statistical multiplexing of spatial loads," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 3044–3058, 2017.
- [21] K. Nichols and V. Jacobson, "Controlling Queue Delay," *Queue*, vol. 10, no. 5, pp. 20:20–20:34, May 2012.
- [22] 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.
- [23] R. Kumar, A. Francini, S. Panwar, and S. Sharma, "Dynamic control of RLC buffer size for latency minimization in mobile RAN," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2018, pp. 1–6.
- [24] T. Herbert. (2011, Nov.) Byte queue limits. [Online]. Available: <https://lwn.net/Articles/469652/>
- [25] L. Kleinrock, "Internet Congestion Control Using the Power Metric: Keep the Pipe Just Full, But No Fuller," *Ad Hoc Networks*, pp. 142–157, November 2018.
- [26] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond Throughput: A 4G LTE Dataset with Channel and Context Metrics," in *Proceedings of the 9th ACM Multimedia Systems Conference*, ser. MMSys '18. New York, NY, USA: ACM, 2018, pp. 460–465.
- [27] F. Baker and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management," Internet Requests for Comments, RFC Editor, RFC, Jul. 2015.
- [28] IEEE, "IEEE Standard for Local and Metropolitan Area Network–Bridges and bridged networks," *IEEE 802.1Q-2018*, July 2018.

- [29] T. Høiland-Jørgensen, M. Kazior, D. Täht, P. Hurtig, and A. Brunstrom, "Ending the Anomaly: Achieving Low Latency and Airtime Fairness in WiFi," in *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. Santa Clara, CA: USENIX Association, 2017, pp. 139–151.
- [30] P. Goyal, M. Alizadeh, and H. Balakrishnan, "Rethinking congestion control for cellular networks," in *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XVI. New York, NY, USA: Association for Computing Machinery, 2017, p. 29–35.
- [31] 3GPP, "Interface between the Control Plane and the User Plane nodes," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 29.244, Jun. 2019, version 16.0.0.
- [32] J. Costa-Requena, A. Poutanen, S. Vural, G. Kamel, C. Clark, and S. K. Roy, "SDN-Based UPF for Mobile Backhaul Network Slicing," in *2018 European Conference on Networks and Communications (EuCNC)*, June 2018, pp. 48–53.
- [33] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin, K. Wen, K. Kim, R. Arora, A. Odgers, L. Contreras, and S. Scarpina. MEC in 5G networks. [Online]. Available: [https://www.etsi.org/images/files/ETSIWhitePapers/etsi\\_wp28\\_mec\\_in\\_5G\\_FINAL.pdf](https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf)
- [34] 3GPP, "NR, Packet Data Convergence Protocol (PDCP)," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.323, Oct. 2020, version 16.2.0.
- [35] J. Gettys, "Bufferbloat: Dark Buffers in the Internet," *IEEE Internet Computing*, vol. 15, no. 3, pp. 96–96, May 2011.
- [36] J. Corbet. (2012, Jul.) Tcp small queues. [Online]. Available: <https://lwn.net/Articles/506237/>
- [37] S. Abbasloo, Y. Xu, and H. J. Chao, "C2TCP: A Flexible Cellular TCP to Meet Stringent Delay Requirements," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 4, pp. 918–932, 2019.
- [38] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

- [39] W. chang Feng, K. G. Shin, D. D. Kandlur, and D. Saha, "The BLUE active queue management algorithms," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 513–528, Aug 2002.
- [40] S. Kunniyur and R. Srikant, "Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 123–134.
- [41] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled Delay Active Queue Management," Internet Requests for Comments, RFC Editor, RFC 8289, Jan. 2018.
- [42] Openwrt. [Online]. Available: <https://openwrt.org/>
- [43] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the tcp congestion avoidance algorithm," *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 3, p. 67–82, Jul. 1997.
- [44] R. Pan, P. Natarajan, F. Baker, and G. White, "Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem," RFC 8033, Feb. 2017. [Online]. Available: <https://rfc-editor.org/rfc/rfc8033.txt>
- [45] P. Imputato, S. Avallone, M. P. Tahiliani, and G. Ramakrishnan, "Revisiting design choices in queue disciplines: The pie case," *Computer Networks*, vol. 171, p. 107136, 2020. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1389128619313775>
- [46] C. A. Grazia, N. Patriciello, T. Høiland-Jørgensen, M. Klapez, M. Casoni, and J. Mangues-Bafalluy, "Adapting TCP Small Queues for IEEE 802.11 Networks," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2018, pp. 1–6.
- [47] P. E. McKenney, "Stochastic Fairness Queueing," in *Proc. of IEEE Int. Conf. on Computer Communications*, June 1990, pp. 733–740 vol.2.
- [48] M. Shreedhar and G. Varghese, "Efficient Fair Queueing Using Deficit Round Robin," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Com-*

- puter Communication*, ser. SIGCOMM '95. New York, NY, USA: ACM, 1995, pp. 231–242.
- [49] “The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm,” RFC 8290, Jan. 2018.
- [50] B. Briscoe, K. D. Schepper, M. B. Braun, and G. White, “Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service:,” Internet Engineering Task Force, Internet-Draft draft-ietf-tsvwg-l4s-arch-06, Mar. 2020, work in Progress. [Online]. Available: <https://tools.ietf.org/html/draft-ietf-tsvwg-l4s-arch-06>
- [51] J. Henry, T. Szigeti, and L. M. Contreras, “Diffserv to QCI Mapping,” Internet Engineering Task Force, Internet-Draft draft-henry-tsvwg-diffserv-to-qci-04, Apr. 2020, work in Progress.
- [52] L. Kleinrock, “Power and Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications,” in *Conference Record, International Conference on Communications*, Boston, Massachusetts, June 1979, pp. 43.1.1–43.1.10.
- [53] J. Jaffe, “Flow Control Power is Nondecentralizable,” *IEEE Transactions on Communications*, vol. 29, no. 9, pp. 1301–1306, Sep. 1981.
- [54] L. Zanzi, V. Sciancalepore, A. Garcia-Saavedra, H. D. Schotten, and X. Costa-Perez, “LACO: A Latency-Driven Network Slicing Orchestration in Beyond-5G Networks,” 2020.
- [55] Google stadia. [Online]. Available: <https://stadia.google.com/>
- [56] S. Abbasloo, Y. Xu, H. J. Chao, H. Shi, U. C. Kozat, and Y. Ye, “Toward optimal performance with network assisted TCP at mobile edge,” in *2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19)*. Renton, WA: USENIX Association, Jul. 2019. [Online]. Available: <https://www.usenix.org/conference/hotedge19/presentation/abbasloo>
- [57] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, “A Survey on Low Latency Towards 5G: RAN, Core Network and Caching Solutions,” *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3098–3130, 2018.
- [58] Latency Numbers Every Programmer Should Know. [Online]. Available: <https://gist.github.com/jboner/2841832/>

- [59] A. K. Paul, H. Kawakami, A. Tachibana, and T. Hasegawa, "An AQM based congestion control for eNB RLC in 4G/LTE network," in *2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, May 2016, pp. 1–5.
- [60] S. Jung, J. Kim, and J. Kim, "Intelligent Active Queue Management for Stabilized QoS Guarantees in 5G Mobile Networks," *IEEE Systems Journal*, pp. 1–10, 2020.
- [61] R. Kumar, A. Francini, S. Panwar, and S. Sharma, "Design of an enhanced bearer buffer for latency minimization in the mobile ran," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.
- [62] "SAEGW Administration Guide," [https://www.cisco.com/c/en/us/td/docs/wireless/asr\\_5000/21-8\\_6-2/SAEGW-Admin/21-8-SAEGW-Admin/21-8-SAEGW-Admin\\_chapter\\_0111010.html](https://www.cisco.com/c/en/us/td/docs/wireless/asr_5000/21-8_6-2/SAEGW-Admin/21-8-SAEGW-Admin/21-8-SAEGW-Admin_chapter_0111010.html).
- [63] M. Ihlar, A. Nazari, and R. Skog. (2018) Low latency, high flexibility - Virtual AQM. [Online]. Available: <https://www.ericsson.com/en/ericsson-technology-review/archive/2018/virtual-aqm-for-mobile-networks>
- [64] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," Internet Engineering Task Force, Internet-Draft draft-ietf-quic-transport-34, Jan. 2021, work in Progress.
- [65] R. R. Stewart, J. Stone, and D. Otis, "Stream Control Transmission Protocol (SCTP) Checksum Change," RFC 3309, Sep. 2002.
- [66] P. Imputato, N. Patriciello, S. Avallone, and J. Manges-Bafalluy, "Smart Backlog Management to Fight Bufferbloat in 3GPP Protocol Stacks," in *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan 2019, pp. 1–6.
- [67] N. Bouacida and B. Shihada, "Practical and Dynamic Buffer Sizing Using LearnQueue," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1885–1897, Aug. 2019.
- [68] M. Irazabal, E. Lopez-Aguilera, and I. Demirkol, "Active Queue Management as Quality of Service Enabler for 5G Networks," in *2019 European Conference on Networks and Communication (EuCNC)*, April 2019, pp. 1–5.

- [69] T. Herbert. (2011, Nov.) Dynamic queue limit. [Online]. Available: <https://lwn.net/Articles/469651/>
- [70] L. Kleinrock, *Queuing Systems*. Wiley Interscience, 1975, vol. 1.
- [71] 3GPP, "NR; Physical layer procedures for data," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.214, Sep. 2019, version 15.7.0.
- [72] —, "NR, Physical channels and modulation," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.211, Jun. 2019, version 15.6.0.
- [73] M. Irazabal, E. Lopez-Aguilera, I. Demirkol, and N. Nikaiein, "Dynamic Buffer Sizing and Pacing as Enablers of 5G Low-Latency Services," *IEEE Transactions on Mobile Computing*, 2020.
- [74] VoIP bandwidth consume. [Online]. Available: <https://www.cisco.com/c/en/us/support/docs/voice/voice-quality/7934-bwidth-consume.html>
- [75] M. Irazabal, E. Lopez-Aguilera, I. Demirkol, R. Schmidt, and N. Nikaiein, "Preventing RLC Sojourn Delays in 5G," *IEEE Access*, 2021.
- [76] X. Che and B. Ip, "Packet-level traffic analysis of online games from the genre characteristics perspective," *J. Network and Computer Applications*, vol. 35, pp. 240–252, 01 2012.
- [77] M. Belshe, R. Peon, and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)," Internet Requests for Comments, RFC Editor, RFC 7540, 2015.
- [78] 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 8290, Jan. 2018.
- [79] OpenAirInterface Allianz. [Online]. Available: <https://www.openairinterface.org/>
- [80] N. Nikaiein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "OpenAir-Interface: A Flexible Platform for 5G Research," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, p. 33–38, Oct. 2014.
- [81] IRTT (Isochronous Round-Trip Tester). [Online]. Available: <https://github.com/heistp/>



irtt

- [82] 3GPP, "NR; Physical layer procedures for control," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.213, Jul. 2020, version 16.2.0.
- [83] C. Hornig, "A Standard for the Transmission of IP Datagrams over Ethernet Networks," Internet Requests for Comments, RFC Editor, RFC, Apr. 1984.
- [84] IEEE, "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 Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pp. 1–3534, 2016.
- [85] 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 *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 427–441.
- [86] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 5th edition. Springer, 2016.
- [87] D. S. Johnson and K. A. Niemi, "On knapsacks, partitions, and a new dynamic programming technique for trees," *Mathematics of Operations Research*, vol. 8, no. 1, pp. 1–14, 1983.
- [88] DBS3900 Huawei Distributed Base Stations. [Online]. Available: <https://e.huawei.com/en/products/wireless/elte-trunking/network-element/dbs3900>
- [89] 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 Transactions on Networking*, vol. 1, no. 3, pp. 344–357, 1993.
- [90] M. H. Goldwasser, "A survey of buffer management policies for packet switches," *SIGACT News*, vol. 41, no. 1, p. 100–128, Mar. 2010.
- [91] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

- [92] W. A. Aiello, Y. Mansour, S. Rajagopalan, and A. Rosén, “Competitive queue policies for differentiated services,” *Journal of Algorithms*, vol. 55, no. 2, pp. 113 – 141, 2005.
- [93] T. Høiland-Jørgensen, C. A. Grazia, P. Hurtig, and A. Brunstrom, “Flent: The flexible network tester,” in *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*, ser. VALUETOOLS 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 120–125.
- [94] M. A. Bender and H. Hu, “An adaptive packed-memory array,” *ACM Trans. Database Syst.*, vol. 32, no. 4, p. 26–es, Nov. 2007.
- [95] L. Arge, M. A. Bender, E. D. Demaine, B. Holland-Minkley, and J. I. Munro, “Cache-oblivious priority queue and graph algorithm applications,” in *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, ser. STOC '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 268–276.