TC-RAN: A Programmable Traffic Control Service Model for 5G/6G SD-RAN

Mikel Irazabal and Navid Nikaein

***Abstract*— Driven by the key principles of open interfaces, virtualization and programmability, Open RAN has emerged as a new paradigm to evolve contemporary Radio Access Networks (RANs) into a more vendor-agnostic, softwarized, and intelligent ecosystem. To this end, Software Defined RAN (SD-RAN) initiatives (e.g., O-RAN) are drafting specifications to provide the means to embrace it. However, even though O-RAN following the Open RAN paradigm specifies Service Models (SMs) to monitor and control the RAN, it does not go beyond the Quality of Service (QoS) mechanisms provided by 3GPP. Therefore, the QoS degradation that occurs mostly due to data flow's nature at the slowest data path link (e.g., high L2 sublayers), is not addressed by contemporary O-RAN SMs. In this paper, we present a traffic control system for SD-RAN, denoted as TC-RAN, that consists of an E2 service Model (E2SM) and a RAN Function (RF) that adheres to the Open RAN principles and promotes data flows to first-class citizens in cellular networks, upgrading contemporary 5G QoS mechanism. TC-RAN introduces a 6 programmable, extendable, and customizable pipeline composed of a classifier, a policer, a queue, a scheduler, a shaper and a pacer. Additionally, TC-RAN addresses QoS degradation scenarios unsolvable through Resource Block (RB) allocation or 3GPP slicing mechanisms, unleashing the true potential for deploying extremely demanding applications and creating a green field for AI/ML cross-optimization algorithms on the road to 6G. We prototype and validate TC-RAN in a real 5G Stand Alone (SA) RAN stack using an O-RAN compatible near Real-Time Radio Intelligent Controller (nearRT-RIC), xApps, and Commercial off-the-shelf (COTS) User Equipments (UEs). The results show that intelligently composing a TC-RAN pipeline in cellular networks can considerably reduce the latency, notably enhancing the Quality of Experience (QoE) in a real multiplayer online game.**

***Index Terms*— Open RAN, O-RAN, E2SM, nearRT-RIC, 5G QoS.**

I. Introduction

OPEN RAN is emerging as a new paradigm for open, intelligent, programmable and virtualized cellular networks. Intelligent networks can forecast the network status, retrieving Key Performance Indicators (KPI) and act accordingly through closed-loop data-driven control algorithms to optimize the performance. To this end, networks require open and standardized interfaces to permit interoperability among different components, and thus, enable vendor-agnostic intelligent algorithms. Additionally, networks entail programmability, unleashing the potential for algorithms to dynamically change RAN's behavior/composition and smoothly adapt to the new demands. Another pillar in Open RAN is virtualization that allows the cloudification of the network, facilitating deployments and dynamically managing resources based on the application demands. Towards these ends, O-RAN [1] is a vendor and operator driven community that drafts standardized protocols which permit interoperability between different vendors, unlocking the SD-RAN paradigm. Among other things, O-RAN defines the E2 Application Protocol (E2AP), which encapsulates the Service Model (SM), providing a standard mechanism of communication between an xApp and an RF. In this way it provides forward compatibility and decoupling, as new unforeseen SMs that address novel challenges can be designed and transported using the E2AP, similar to how IP packets agnostically transport L4 protocols (e.g., TCP or UDP) in their data section. One of these novel challenges that are still not tackled by O-RAN or 3GPP, and that can be designed using the Open RAN principles and E2AP/SMs is the *RAN data flow control*.

As shown in Fig. 1, the throughput and Round Trip Time (RTT) of two contemporary softwarized 5G RAN stacks in SA mode under the Real-Time Response Under Load (RRUL) test provided by flent [2], present delays in the order of hundreds of milliseconds, which exceeds the latency envisioned by 3GPP for low-latency scenarios [3]. During the first 5 seconds in the RRUL test, only the UDP and ping flows are active, and thus, we can observe that the no-load delay of the system is around 10 ms, a suitable value for time sensitive over the top (OTT) applications [4]. However, after the 5th second, the TCP flows begin and the perceived delay increases dramatically. Even worse, in Fig. 1 the reported results were generated using a non-loss-based Congestion Control Algorithm (CCA) (i.e., TCP BBR [5]), which aim to achieve low latency, estimating the available throughput and pacing the traffic accordingly. In fact, the observed downward spikes in the total throughput in Fig. 1 happen every 10 seconds during 200 ms[1] where the CCA tries to estimate the available bandwidth in a no-load scenario. Moreover, even though the flows are marked with different QoS classes (i.e., DiffServ Best Effort (BE), Background (BK), Signaling (CS5) and Expedited Forwarding (EF)), they end up sharing a common queue (i.e., Radio Link Control (RLC)) and packets that are time sensitive (e.g., EF is used for sensitive real-time, interactive traffic)

Manuscript received 7 January 2023; revised 29 May 2023; accepted 2 August 2023. Date of publication 28 November 2023; date of current version 17 January 2024. This work was supported in part by the European Commission as a part of the H2020 Program 6GBrain Project under Grant 101017226 and in part by the Horizon Europe 2022 ImagineB5G Project under Grant 101096452. *(Corresponding author: Mikel Irazabal.)*

Mikel Irazabal resides in 34121 Trieste, Italy (e-mail: mikel@irazabal.eu).

Navid Nikaein is with the Department of Communication Systems, Eurecom, 06410 Sophia Antipolis, France (e-mail: navid.nikaein@eurecom.fr).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2023.3336162.

Digital Object Identifier 10.1109/JSAC.2023.3336162

[1]From the linux kernel net/ipv4/tcp_bbr.c.

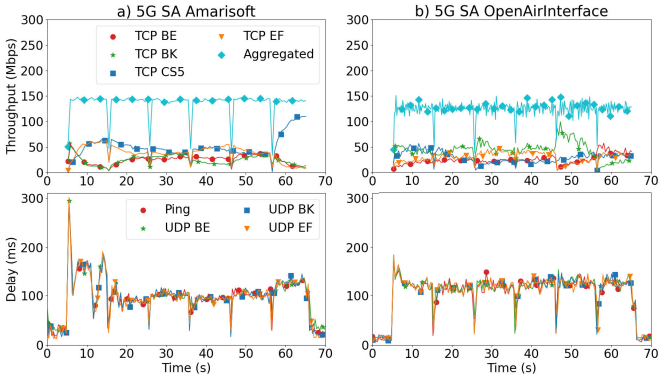

Fig. 1. RRUL in Amarisoft and OpenAirInterface (OAI) 5G SA RANs with TCP BBR enabled in both endpoints.

suffer considerable latency, ruining user QoE. This outcome clearly shows the necessity to upgrade the current 5G QoS mechanism.

To meet latency requirements, 5G proposes slicing, that has been traditionally presented as a key enabler for low-latency communications [6], [7]. When a slice is created, a new PDU Session is established [3], and flows marked with the appropriate QoS Flow Indicator (QFI) are redirected to the newly created Data Radio Bearer (DRB) through the Service Data Adaptation Protocol (SDAP) sublayer. O-RAN envisions the necessity to dynamically map the QFI to DRB, and thus, provides an SM (i.e., E2 RAN Control (E2RC) [8]) that permits its control using an xApp. In fact, 3GPP's QoS mechanism is very similar to the DiffServ mechanism. Packets are classified into classes (e.g., real-time gaming), and segregated in different queues, from which a scheduler dequeues them every transmission opportunity. However, correctly segregating the flows is challenging due to: (i) the growing privacy and security concerns (e.g., Tor project or VPNs), (ii) the cloudification of the services, and thus, network dynamicity, and (iii) the lack of communication among the service providers and the content providers. Therefore, there exists a considerable amount of packets that may be time sensitive that will end sharing the default DRB, rather than instantiating a new dedicated bearer. Moreover, even if the flows can be segregated, if two flows from different services belong to the same QFI, they will share the RLC buffer and 3GPP does not provide any fairness mechanism to avoid one greedy flow to monopolize the access to the resources. Hence, the latency requirements of the second flow will be ruined, as its packets will have to wait in a bloated queue until the packets of the former have been transmitted, a situation that can be observed in Fig. 1. Additionally, 3GPP's QoS mechanism is a funnel where $[1, 64]$ QFIs per UE can exist, while only $[1, 32]$ DRBs can be instantiated [3], [9], worsening the situation due to the pigeonhole principle. Lastly, 3GPP QoS finest grained mechanism is QFI, and hence, there does not exist a data flow slicing possibility within the standard framework for flows marked with the same QFI.

State-of-the-art solutions for this problem in the wired and Wi-Fi stack suggest using Active Queue Management (AQM) and Fair Queuing (FQ) mechanisms [10], [11] [12]. However, in a 3GPP network, packets marked with the same QFI should be treated equally [3], and thus, no fairness mechanism among flows is envisioned. Additionally, AQM techniques where packets may be dropped (e.g., using CoDel) or further segregated do not strictly comply with 5G QoS Model. Therefore, other techniques that mark the Explicit Congestion Notification (ECN) bit, to reduce senders' pace have recently being adopted by 3GPP [3] (i.e., Low Latency, Low Loss, Scalable Throughput (L4S) [13]), trying to tackle the sojourn latency problem generated by greedy flows.

Other 3GPP limitation is the fact that the QFI classification is performed in the UPF, which may be located geographically distant from the RAN depending on the type of service, as shown in Figure 2 (i.e., at the edge, regional, or central cloud). Such placement intrinsically brings an associated latency. This calls for a native flow-based QoS differentiation in the RAN user-plane to support stringent requirements of forthcoming services in terms of latency and reliability, and entails further architectural evolution in enabling RAN to natively process data flows.

On account of the aforementioned shortcomings, in this paper we propose and validate a programmable SD-RAN data plane pipeline that promotes data flows to first-class citizens in the RAN beyond the 3GPP specification in twofold: (i) introducing a new RF and a new O-RAN compatible SM, and (ii) implementing a concrete instance of TC-RAN using a real 5G SA testbed, COTS UEs, xApps, and an O-RAN compatible nearRT-RIC to validate our approach.

To this end, the main contributions of this paper can be summarized as follows:

- Introduce a programmable, composable, extendable, customizable SD-RAN data flow processing pipeline to natively treat user data packets as first-class citizens in 5G adhering to the Open RAN principles.
- Present an architectural upgrade in 5G's QoS mechanism, enhancing the granularity from QFIs to data flows, and thus, enabling data flow control, that works in near real-time.
- Validate TC-RAN in a real 5G SA testbed using an O-RAN compatible nearRT-RIC with COTS UEs and xApps, and provide a concrete implementation of it.
- Contribute to the research community with an extendable tool to manipulate data flows in the RAN with 5G/6G specificities (e.g., 5G-BDP pacer).

The remainder of the paper is structured as follows. In Section II we describe the related work regarding QoS in 5G and the Open RAN contributions. Afterwards, in Section III we give an overview of the proposed TC RF/SM. The following Section IV is devoted to provide a detailed architectural overview, and in Section V we show the results that we obtained while validating TC-RAN. This paper outlines the conclusion in Section VI.

II. Related Work

This section can be divided between the Open RAN contributions and the contributions regarding the QoS model in 5G.

A. 5G's QoS Model Limitations

There has been a significant QoS enhancement in 5G aiming to guarantee latency and throughput, and thus, unlock

TABLE I
MOST COMMON TERMINOLOGY USED IN THIS PAPER

Abbrev.	Meaning
E2	O-RAN-defined interface
E2AP	Application Protocol for E2 management and SM encapsulation
E2SM, SM	Service Model for information exchange
E2TC, TC	E2 Traffic Control
nearRT-RIC	near Real-Time RAN Intelligent Controller
RAN	Radio Access Network
RF	RAN Function. Managed via E2SM
SD-RAN	Software-Defined Radio Access Networking
xApp	eXternal App. Manages a RF via an E2SM

the potential for novel applications to emerge. Following, we explain the data path for a packet in downlink while similar reasoning applies for uplink due to 5G's stack symmetry. In 5G, data packets firstly arrive to the User Plane Function (UPF), which supports 5 different type of PDU sessions (i.e., IPv4, IPv6, IPv46, Ethernet or unstructured) [3]. At the UPF, among other things, packets are segregated through a Packet Detection Rule (PDR), policed using the Multi-Access Rule (MAR), forwarded based on the Forwarding Action Rules (FARs), tagged based on the QoS Enforcement Rules (QERs) and lastly reported employing the Usage Reporting Rules (URRs). Packets are egressed from the UPF marked with a Qos Flow Indicator (QFI), which is the finest grain quality indicator provided by 3GPP. Besides, a new RAN sublayer has been introduced (i.e., SDAP [14]), which principal *raison d'être* is to map the QFIs into the DRBs. The next stage in the packet journey is found in the RLC buffer, where packets sojourn until they are scheduled by the MAC sublayer and forwarded to the PHY layer. This QoS model based in QFIs is very similar to DiffServ,[2] where every class has associated some QoS features, and the MAC scheduler decides which packets to schedule next, based on different policies.

However, 5G's QoS model is based on packet tagging, which is becoming more challenging in contemporary networks as the privacy and security concerns increase (e.g., Tor Browser or VPN connections), where the information needed for classification (e.g., 5-tuple) may not always be available to the UPF. Moreover, contemporary applications' cloudification trend amplifies the problem. Applications may dynamically change their IP address while migrating, searching to increase or shrink their resources as needed, effectively limiting UPF's capability to tag the packets. Furthermore, 5G's QoS model assumes information exchange between the applications content providers and the network service providers for correctly tagging the packets, which may not always occur in a system with increasingly OTT applications. In such cases, flows from diverse applications shall be tagged with the same QFI, and thus, they will share the last buffer (i.e., RLC in downlink) before the slowest data link. There, packets are handled in a FIFO manner, and thus, one greedy flow can monopolize the queue, negatively affecting the remaining flows tagged with the same QFI, since no contention or fairness mechanism is foreseen. In fact, from the MAC scheduler's perspective there exists a group of different queues with diverse QFIs, where the ordering of arrival within the queues needs to be respected, forming a partially ordered set or poset [15]. Additionally, the MAC scheduler lacks any ability to segregate the packets within an RLC queue, as their contents may have been ciphered for security reasons at the Packet Data Convergence Protocol (PDCP). Therefore, no further fine-grained processing is possible such as fairly scheduling the packets belonging to the same QFI that were generated from different applications, or isolating and dropping packets from abusive sources. Other 5G's QoS model shortcoming happens due to the funnel nature of the 3GPP stack (i.e., 64 QFIs and 32 DRBs), where different flows will share RLC buffers in the data path due to the pigeonhole principle even if tagged with different QFIs. In all the aforementioned situations, data flows with diverse requirements may end sharing the radio bearer, and thus, the RLC buffer. In such situations, it is not possible to address the problem from the MAC scheduler, where the order of arrival in the queues needs to be respected, and thus, 5G's QoS model limitations to tackle the root causes become tangible.

Moreover, the situation is exacerbated when packets are transported using a loss-based TCP algorithm (e.g., Cubic [16]), where the sender's pace is estimated through packet losses, and large queues are formed before the slowest link in the data path. This behavior could be avoided, limiting the traffic of abusive flows at the UPF. However, correctly estimating the cellular network throughput is very challenging, and thus, the optimal flow's throughput, since queues need to be maintained full to prevent RBs starvation while every extra packet just adds extra sojourn time [17]. In fact, it has been theoretically proved that it exists an optimal pacing point to forward packets in a network [18], achieving full bandwidth utilization and minimum delay. Unfortunately, it has also been proved that no distributed congestion control algorithm (CCA) converges to the aforementioned optimal pacing point [19]. Besides, transmitting packets optimally in cellular networks is very challenging due to: (i) the variability in the wireless quality channel, and consequently in the available bandwidth (e.g., fading due to user mobility), (ii) occurrence of high packet losses, (iii) unpredictability in the number of active users and their performance requirements (e.g., URLLC), and (iv) unknown vendor-specific MAC scheduler algorithm. Hence, it is very difficult to estimate the correct traffic flow's throughput, a fact that it is aggravated due to the distance, and thus the latency, between the UPF and the RLC buffer (see Fig. 2).

B. Open RAN

The advantages that abstractions provide separating the interface from implementation details have been largely proven, being the C programming language probably the most notable example [20]. In the network engineering discipline, OpenFlow [21] emerged as a standardized interface for Ethernet switches. More recently, P4 [22] appeared as a programming protocol for processing packets, which has already been proven in 5G contexts [23]. Similarly, in the 5G domain, Open RAN aims to define common RAN interfaces to provide multivendor interoperability. In this context, O-RAN is an alliance [1] that provides standardized interfaces (e.g., E2

[2]DiffServ and QFI use 6 bits i.e., $2^6 = 64$ classes.

or A1) to achieve such goals. O-RAN defines a near-RT RIC which is responsible through SMs to monitor information from E2 Nodes (e.g., DU, CU or gNodeB) and control resources. At the time of writing this manuscript, O-RAN defines 4 SMs: the RAN Function Network Interface (NI) [24], the Key Performance Metric (KPM) [25], the RAN Control (RC) [8] and the Cell Configuration and Control (CCC) [26]. The NI SM exposes the Network Interfaces, while the KPM SM is used to monitor statistics from the E2 Nodes. RC SM defines the interface to control the Radio Bearers, map the QoS flows to DRBs, the Radio Resource Allocation, the Connected Mode Mobility Control, the Radio Access Control, the Dual Connectivity or the Carrier Aggregation, among other things. Lastly, the CCC SM can be used to control and monitor the configuration of the cells. However, O-RAN does not define any method within any SM to manipulate the data flows beyond mapping the QFI to DRB, limiting itself to the current 5G QoS model. Other SMs based in 3GPP sublayers have also flourished [27], where, similarly to the KPM SM, statistics can be gathered from the E2 Node based in the sublayer (e.g., MAC, RLC, PDCP). Regarding Open RAN ML/AI solutions, other works have focused on the benefits of using Open RAN ML/AI solutions [28], [29] to control and optimize the RAN, without going beyond 5G's QoS mechanisms.

In summary, none of the Open RAN solutions known to the authors provide a remedy to tackle the per-flow 5G QoS problem in cellular networks.

C. 5G QoS in the Research Community

A considerable number of research papers try to improve 5G's QoS optimizing the system operating point (i.e., achieving low latency and full throughput) in cellular networks through the CCA [5], [30], trying to apply Kleinrock's motto of *keep the pipe just full, but not fuller* [17]. If the sender transmits packets at a higher rate than the link capacity of the slowest link in the path, packets start accumulating (e.g., in the RLC sublayer in 5G), bloating the buffer and ruining any latency requirements of other flows that share the link with the greedy flow. Additionally, this effect is accentuated if the CCA is loss-based (e.g., Cubic [16]) as it estimates the link capacity through packet losses, which occur at the buffer before the slowest link in the path once it is bloated. In fact, the impact of bloating the buffers (i.e., bufferbloat) in the latency on cellular networks has been known by the research community by at least a decade [31]. Tackling the problem from a sender's perspective resolves it in specific situations, but cannot avoid the notification delay from the congestion location (i.e., RLC buffer) until the sender's CCA. Additionally, CCA cannot gather RAN status information to optimize the pace. Therefore, some new CCA based on deep reinforcement learning techniques [32] try to guess the optimal point at which each flow should be transmitted, trying to anticipate the network status when packets are forwarded. However, all the CCA solutions try to address the problem from a single flow's perspective, but they lack the ability to group different flows and assign them specific semantic (e.g., they all belong to one application), or steer the traffic. Moreover, a CCA solution lacks the capacity to isolate and limit the bandwidth of a greedy competing flow.

Other approaches try to estimate the rate and the pace of the packets positioning themselves before the 3GPP stack [33], [34] treating the RAN as a black box, and backlogging the packets. Indeed, algorithms located closer to the congestion point will inherently react faster, and thus, reduce the latency while trying to fully utilize the available throughput. However, this approach will also miss any information that can be leveraged from the 3GPP stack to improve the performance.

Another traditional approach to improve the QoS from a flow's perspective is to directly act in the elements involved in the data path (e.g., using Active Queue Management (AQM)). In fact, the 10th anniversary for the first implementation of the nowadays, well understood and deployed AQM algorithms CoDel and FQ-CoDel [35] has been recently celebrated. The FQ-CoDel algorithm is based in a Deficit Round Robin (DRR) scheduler and a group of queues, where the flows' packet header is hashed, and the packets are enqueued accordingly. Each queue has a quantum to guarantee their fairness when scheduled in a Round Robin (RR) manner, as well as a Codel AQM to keep the latency and the queue size under control. This approach has been explored in Wi-Fi wireless communications with excellent results [11], [36], where additionally to these techniques, the mature Linux network stack provides the versatile TC utility to handle the flows. However, apart from a few exceptions [15], [37] [38] with limited adoption but encouraging outcomes, these mature solutions have not been widely applied in cellular networks.

Recently, L4S [13] has been proposed and adopted in 3GPP [3] as a remedy to fight the bufferbloat, segregating the flows in "classic" (i.e., flows with propensity to bloat the buffers such as Reno or Cubic) and "scalable" (e.g., TCP Prague [39]), and forwarding them in two different queues [40]. In case that congestion is detected, packets are marked when entering the PDCP sublayer with the Explicit Congestion Notification (ECN) bit, signaling the receiver to reduce its pace with promising results [41]. However, L4s does not address the fairness problem of different flows marked with the same QFI beyond the two queues approach.

Other interesting approaches to improve the QoS mechanism focus in the RBs allocation problem through network slicing [42], [43], predicting the network status and acting beforehand to optimize the resources. However, these approaches do not consider the dynamicity of the network (e.g., a delay delivering a TCP packet may alter the CCA transmission rate, and thus, the overall test) and cannot handle the per-flow QoS problem or are not 3GPP standard.

In summary, TC-RAN similarly to the CCA or black box approach, works in per-flow level, but contrary to them, it can directly leverage the RAN status information through an xApp, and thus, it has better chances to optimize the overall performance. Additionally, it has information of all the current flows in the RAN, contrary to the CCA, that can only infer the network status. On the other hand, TC-RAN is flow based, and thus, it provides solutions unsolvable through a MAC slicing algorithm (e.g., abusive flows, flow level slicing or bufferbloat). Moreover, TC-RAN provides great flexibility through its 6 pipeline stage, which allows composing complex QoS mechanisms, in contrast with L4S type of solutions where the mechanism cannot be dynamically modified. Furthermore,

it handles the QoS problem just before the slowest link in the data path, in contrast to 3GPP's UPF that may be localized far away from the RAN, providing an upgrade for current 5G QFI mechanism. On top of it, TC-RAN is built following the Open RAN principles as an O-RAN SM, embracing open interfaces that permit vendor interoperability, and thus, interaction with other SMs (e.g., RC) for cross-optimization algorithms.

III. Traffic Control Overview

The core contribution of this paper is a novel Traffic Control system that adheres to the Open RAN principles and promotes data flows to first-class citizens in cellular networks to upgrade the overall QoS, as well as a natural architectural evolution to reduce the user plane latency. While latency reduction is one of the main reasons for such architectural evolution, there is more into it: *treating 5G RAN as a native packet Access Point (AP)*. Meeting the low-latency requirements demanded by the Service Level Agreements (SLAs) that cannot be handled by the MAC resource allocation scheduler (e.g., the bufferbloat), while embracing to Open RAN principles (i.e., through O-RAN SMs) is the primary, and yet unsolved challenge in the 5G context, addressed by TC-RAN.

The TC-RAN system consists of the following components: (i) the TC RF, located between the SDAP and the PDCP sublayers, which consists of a 6 stage pipeline and processes the data plane, and (ii) the TC SM, that enables the control plane of the TC RF, both of them depicted in blue in the Fig. 2. There exists one TC RF pipeline per user DRB, and thus, the TC RF can handle, IPv4, IPv6, IPv4v6, Ethernet or Unstructured packets [3]. Similarly to the O-RAN RFs and SMs, TC-RAN is managed by at least one xApp, which monitors and controls the packet processing pipeline and enforces diverse actions to the flows that traverse the TC RF in near real-time. The xApp is connected to the RF through a nearRT-RIC and an E2 Agent, as shown in Fig. 2.

The proposed TC-RAN unlocks the possibility for controlling individual flows arriving from the layer 3/4 protocols (e.g., IP/TCP) through TC-RAN, upgrading 5G QoS mechanism from QFIs to flows. In this way, an xApp can optimize the RAN for diverse purposes such as fairly distributing the RBs among flows marked with the same QFI, isolating abusive flows, or drop a packet to reduce the bufferbloat generated by TCP flows (e.g., CoDel) to meet guaranteed latency constraint, to name a few.

A. Data Plane

As shown in Fig. 2, the TC-RAN RF pipeline is located after the SDAP but before the PDCP sublayer. Since the SDAP maps the packets from their QFIs to DRBs, the packets still preserve their header, and thus, they can be classified accordingly, even though care shall be taken as the SDAP sublayer may include an optional 1 byte header [14]. Furthermore, PDCP may alter the packet header information [44] (e.g., using the Robust Header Compression (ROHC) algorithm) before forwarding the packets to the RLC queue, where packets are buffered until the MAC scheduler requests them, and the PHY layer transmits them. Therefore, the natural place to locate TC-RAN pipeline lies just before the PDCP and after the SDAP sublayer, as shown in Fig. 2. The same reasoning was applied when deploying L4S in 5G for marking the packets [41], leading to a marking mechanism located just before the PDCP. The proposed TC-RAN system processes packets on per user DRB basis, being therefore compatible with F1AP [45] split procedures. In downlink, packets arrive in the RAN from the UPF, where they are marked with a QFI according to their requirements. Once in the SDAP sublayer, the packet is mapped to the appropriate RLC DRB according to the QFI. In the first stage, a CLASSIFIER maps the packet to a queue according to the classifier's configuration. Examples of classifiers are Deep Packet Inspectors, where packets are dispatched according to L3, L4 and L7 information, or hash based classifiers (e.g., Stochastic Fair Queuing [46]), where a packet may be redirected according to its 5-tuple (i.e., source address, destination address, source port, destination port and protocol). Once a packet has been classified, a POLICER may accept forwarding the packet, decide to deviate the packet to another queue due to excessive traffic detected through a meter or even drop it. If the allowed rate has not been surpassed, the packet is forwarded to the third stage, where a QUEUE buffers them. Different types of queues (e.g., CoDel or FIFO) can coexist within TC RF tailoring different types of traffic (e.g., "classic" vs "scalable" in L4S specification [13]). In the fourth stage, a SCHEDULER decides from which queue the next packet should be dequeued.

Typical schedulers include but are not limited to RR or priority based schedulers. A SHAPER determines in the fifth stage if the packet shall be forwarded or not according to the measured traffic rate. Contrarily to the policer, the shaper has the capability to delay a packet if the desired rate is surpassed. Lastly, a PACER, determines whether the packets are forwarded or delayed. This stage is crucial as it permits avoiding bloating the RLC buffer, and thus avoids the bufferbloat, while it should always provide enough packets to the RLC buffer to fully utilize the available RBs. To ensure the correct functioning of the PACER, feedback from the RLC buffer status is needed, which in monolithic RANs may be directly obtained from the RLC, and in disaggregated RAN's could come through the NR user plane protocol, where the desired buffer size and the data rate per radio bearer is transmitted from the distributed unit (DU) to the centralized unit (CU) using the F1-U interface [45] or through an xApp subscribed to the RLC sublayer in the DU. Cellular network examples of pacers are DQL or 5G-BDP [47], where the backlog mechanism applies. Contrarily to the shaper, the pacer considers the rate of all the queues within the SM, rather than the rate of a single queue.

B. Control Plane

The TC SM is designed to be transported using the O-RAN E2AP protocol, and thus, facilitate its future integration into different RAN vendors' stack. Additionally, the TC SM exposes the capabilities of the TC RF by means of a set of control action primitives. Moreover, the TC RF can be dynamically composed, customized and reconfigured by an xApp via a TC SM, allowing to define how each packet should be processed. Such an xApp acts as a TC RF controller, and triggers a RIC Control message towards the intended E2 Agent. Upon the reception and decoding of the message, the TC RF updates the pipeline accordingly (e.g., add, modify,

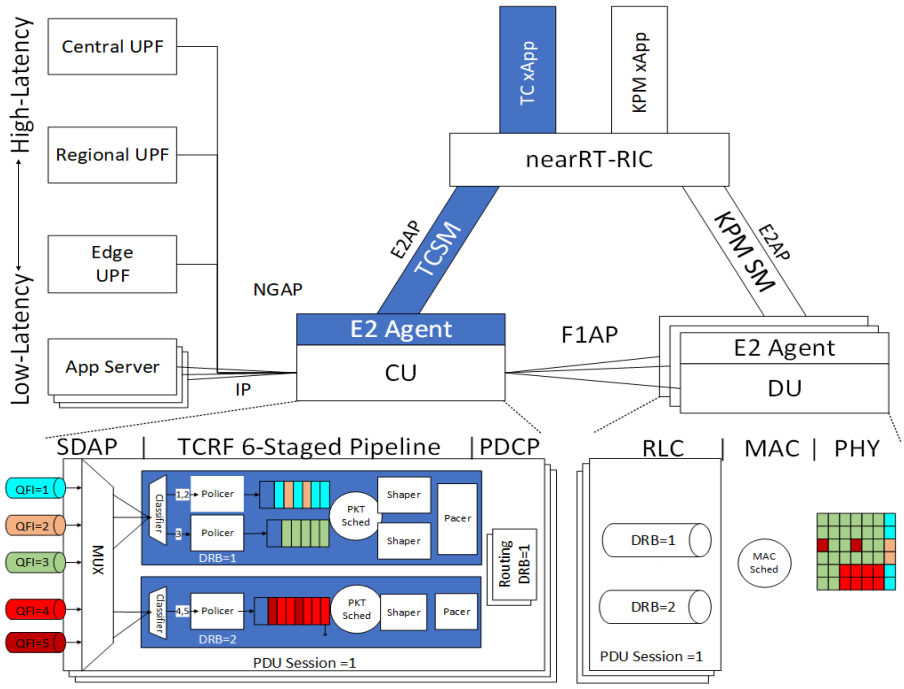

Fig. 2. Proposed TC-RAN system with its pipeline in a disaggregated RAN (i.e., CU-DU) in conjunction with the nearRT-RIC with onboarded xApps and different UPFs. The navy blue parts show the contributions made by TC-RAN.

or delete a component). For example, once detected an abusive flow, an xApp can add a new CoDel queue, add a new filter in the OSI classifier to deviate the traffic belonging to that IP address, modify and limit its accepted rate, and modify the queue's priority when scheduling.

C. Enabling TC-RAN Capabilities in the RAN

5G enables provisioning of UPFs tailored to specific use cases. As shown in Fig. 2, the UPF may be placed in different locations based on the service types and the latency constraints. UPF detects applications using the Service Data Flow (SDF) traffic filter templates, routes and forwards packets, as well as handles per-flow QoS. This mechanism presents limitations as it is far from the bottleneck of the data path (i.e., RLC in downlink), and thus, AQM techniques do not apply (e.g., CoDel) [48]. Moreover, to avoid the bufferbloat, information regarding the channel status from the RAN needs to be retrieved, with its non-negligible delay [47]. Therefore, latency constrained applications (i.e., URLLC [49]) are envisioned to be provisioned with an edge UPF, and converged UPF-CU is the logical evolution of this trend in 6G, treating the data flows directly in the RAN. In this manner, the data path distance is reduced, and thus, the latency. Therefore, the natural architectural evolution for reducing the latency of the data path leads to our proposed design, where the data flows of low latency applications are directly managed by the RAN, avoiding the unnecessary GTP encapsulation/decapsulation that occurs between the UPF and the CU, through the NGAP protocol [50]. This brings user plane (UP) capabilities directly into the RAN, closer to the slowest link in the data path, which is the wireless channel as shown in Section V. Thus, with the proposed TC-RAN system, the RAN acquires new packet processing capabilities, able to directly interact with edge/local applications servers (e.g., for time-sensitive or synchronous applications) without any intervention of the UPF and just before the slowest data path link, avoiding unnecessary delays.

IV. Architecture and Implementation

In this section, we elaborate on the TC-RAN architectural design, challenges, and implementation details, primarily focusing on the following design challenges:

- Design and implement a programmable data flow processing pipeline for 5G/6G SD-RAN that supports composability, customizability and extendability.
- Create a novel set of user-plane abstraction in a form of (open) interfaces between the TC RF and E2 Node, that is applicable to both current and future cellular systems, ensuring backward and forward compatibility.
- Integrate TC-RAN according to the O-RAN specifications, and thus, unleash Open RAN's potential as a vendor-agnostic open and standard solution, cleanly segregating its control plane (CP) and UP through its E2AP protocol.

A. TC RF Flow Processing Architecture Pipeline

The TC RF provides a modular and extensible interface following the Unix philosophy [51], based on six stages: classification, policing, queuing, scheduling, shaping and pacing, that, as observed in Fig. 3, can be separated in two mayor actions: ingress and egress.

1) Ingress::Classifier: The first stage that a packet traverses in the TC RF pipeline is the classifier. The classifier's *raison'être* is to segregate the packets into different queues. The classifier answers the question of where a packet should be routed. Therefore, the input of a classifier is a packet and its output a queue ID. No restriction on the packet header type is applied, accepting current 5G IPv4, IPv6, IPv4/IPv6,

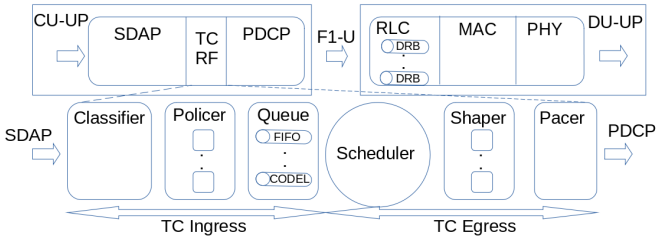

Fig. 3. TC RF pipeline in downlink.

Ethernet and unstructured PDU sessions [3]. However, due to the location of TC RF before PDCP, the packet header information can be used for classification purposes (e.g., the 5 tuple in IP PDU sessions). The classifier also needs to register the queue stage status, specifically the amount of active queues and their ID, which may change dynamically. Only one classifier per RLC buffer exists. The taxonomy algorithm is implementation specific, while typical algorithms segregate the packets into different queues stochastically [46] or using OSI model information. For example, the concrete OSI classifier that we developed can segregate packets according to their Layer 3 and Layer 4 information, allowing a 5-tuple segregation. Thus, packets are separated according to their protocol, source address, source port, destination address and destination port[3] Typical statistics that can be gathered are the amount of packets forwarded to a specific queue, while typical configuration values are based in packet filter match actions. The classifier can be loaded online as it is designed as a shared object, enabling forward compatibility.

2) Ingress::Policer: The second stage in the pipeline is the policer. Once the classifier has assigned the destination queue for the packet, the responsibility moves to the policer. The policer can reject the packet received from the classifier or redirect it to another queue, if a rate limit is reached within a time window. The policer answers the question of how much data rate can be accepted, and thus, it decides either to forward the packet, deviate it or drop it. Typical statistics are the transmitted rate, and the deviation rate. The maximum rate, and the time window for calculating them, can be dynamically configured. Only one policer can exist per queue and its lifetime is tied to the lifetime of the associated queue, and thus their relation is 1:1.

3) Ingress::Queue: The TC RF queue module ingresses and egresses packets after the policer and before the scheduler. A queue is a buffer where packets are gathered before being forwarded to an RLC buffer. The queue does not answer any question as it serves as a buffer between the ingress and the egress actions, even though it can drop a packet if its maximum capacity is reached. Examples of queues are FIFO, CoDel [52] or RED [53]. Typical statistics show the average sojourn time of the packets or the amount of packets accumulated, while typically its maximum capacity can be configured among other parameters. Even though from a theoretical point of view, infinite queues could be added, from a practical point of view at most few dozens of queues are expected. Queues can be dynamically created.

[3]More details can be found at: https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/tc-ran/openair2/tc/cls/osi/osi_cls.c.

4) Egress::Scheduler: Scheduler is the first stage in the egress journey of a packet after the queues. The scheduler decides from which queue the next packet should be dequeued. To draw the decision, the scheduler can gather information from the queues' status for scheduling decisions (e.g., schedule the largest queue or the smallest packet first). Since the packet that is selected from the scheduler may not be egressed due to the refusal of the shaper or the pacer, the scheduler gets notified whether the packet from the queue that it selected was ultimately egressed. TC-RAN's scheduler is designed to be forward compatible.[4] An scheduler needs to implement the interface $queue\_t * (*next\_queue)(structsch\_s*)$, returning the next queue from which the scheduler will pop the following packet. In this manner, internal implementation remains private and forward compatibility is achieved.[5] Examples of schedulers include but are not limited to priority schedulers, where the packets from one queue are preferred, or RR schedulers. Typical statistics show the amount of packets egressed from specific queues, while the algorithm to schedule the queues can typically be configured. Only one scheduler per RLC buffer can be active. Schedulers can also be uploaded "on the fly" as they are implemented as plug-ins.

5) Egress::Shaper: Once the scheduler has decided from which queue the following packet should be egressed, the decision falls in the shaper. The shaper allows egressing a packet from a queue if a rate has not been reached. Therefore, it answers whether a packet should be egressed or not. Similar to the scheduler, the shaper needs to be informed if the packet was ultimately forwarded. Contrarily to the policer, the shaper cannot discard a packet, but it can limit the rate at which packets are egressed. The shaper only answers whether the following packet can be dequeued or not. Typical statistics show the current rate of a queue, which can also be configured, as well as its time window to calculate it. Similarly to the policer, the shaper's lifetime is bound with the lifetime of a queue, and thus, it can only exist if an associated queue exists.

6) Egress::Pacer: The last stage in the TC RF pipeline is the pacer. While the shaper works per TC RF queue, the pacer is responsible for not bloating the RLC buffer that exists before the MAC sublayer, that aggregates the data egressed from all the TC SM queues. The pacer also responds whether the following packet should be egressed or not. Typical statistics are the total bandwidth or the amount of packets forwarded to the RLC, while its parameters to configure are related to the pacing algorithm. Examples of pacers are the DQL or the 5G-BDP [47] which can be dynamically loaded.

B. Programability

Besides, the TC SM can configure the RF 6 stages using the add/delete/modify actions relying on the E2AP Control message generated by the nearRT-RIC, while the statistics are gathered using the E2AP RIC Indication message, similar to other E2SMs (e.g., KPM).

TC SM's pipeline design allows composing complex architectures tailored to specific use cases. To name a few: (i) the

[4]https://gitlab.eurecom.fr/oai/openairinterface5g/-/tree/tc-ran/openair2/tc/sch/sch.h line 16

[5]Linux CCAs use the same technique using function pointers, as it can be observed at linux/net/ipv4/tcp bbr.c.

FQ-CoDel with cellular network specificities can be smoothly composed using a stochastic classifier, CoDel queues, a deficit round-robin scheduler and a 5G-BDP pacer, (ii) or the L4S architecture can be implemented using a classifier that segregates the flows into "classic" and "scalable", two queues capable of marking (i.e., ECN) or dropping the packet when congestion is detected, a conditional priority scheduler and a pacer to avoid bloating the RLC buffer, (iii) or a flow slicing pipeline can be configured deploying an OSI classifier to segregate the flows, a policer to limit the input data rate, FIFO queues, a RR scheduler to evenly allocate transmission opportunities, a shaper to limit one queue's rate and a 5G-BDP pacer to avoid the bufferbloat at RLC. Additionally, these configurations are dynamic. If an anomaly is detected, (e.g., all the packets are classified to the same queue since they are transported inside a VPN, and thus, they share a common flow-identifier or the CoDel sojourn time is not adequate for current network conditions) the pipeline can be reconfigured. This flexibility is achieved using xApps that can subscribe to specific SMs using the E2 Subscription procedure for monitoring the status of the RAN and control it through the E2 Control procedure. In this way, an algorithm (e.g., ML/AI algorithm) in an xApp that is subscribed to the O-RAN RC and KPM E2SMs and to the proposed TC SM can monitor the RAN behavior through the KPM E2SM, perform actions at packet level if necessary through the TC SM (e.g., flow slicing or bufferbloat avoidance), and adjust the bandwidth (i.e., the bandwidth in 5G is highly correlated to the amount of RBs assigned and the Modulation and Coding Scheme (MCS)) utilizing the RC E2SM.

C. TC-RAN Implementation

Regarding the code used in our implementation, the TC-RAN can be divided between the code that is necessary to define its E2SM compatible data layout with its encoding/decoding schemes and the code that is needed in the TC RF. The former and later contain approximately 5K lines of C11 code each (we use the _Generic reserved keyword for static polymorphism in the encoding/decoding scheme, with no external dependencies). TC-RAN has been prototyped in downlink in the 5G gNodeB of OpenAirInterface (OAI) [54].

The TC RF needs to communicate with the SDAP and the PDCP sublayers in downlink to ingress and egress packets as required. However, the API needs to be as small as possible yet complete. To identify the RLC DRB, the RAN, needs to pass the Radio Network Temporal Identifier (RNTI), as well as the DRB ID, which uniquely identifies the RLC buffer, in conjunction with the packet address location and its size. For egressing, a mechanism periodically traverses the active TCs, activating the pipeline and forwarding packets if required. TC-RAN is implemented using the scalable I/O system call notification facility epoll, and a configurable timer file descriptor. Since different TC RF instances do not share data among them, if further parallelism is needed, different instances could be processed in parallel. Thus, ingress and egress can be considered as two independent processes that only share a queue. We prototyped TC-RAN in a monolithic gNodeB where we obtained the RLC buffer status periodically, just after the MAC scheduler has formed the TBS, and thus, the packets from the RLC buffer have been dequeued. However, the RLC buffer status needed for the pacer, could also be gathered using an xApp or in a disaggregated deployment, this can also be achieved through the F1-U protocol Downlink Data Delivery Status procedure messages [45], where the desired buffer size and its data rate for the corresponding data radio bearer is transmitted.

TABLE II
NORMALIZED CPU AND MEMORY USED

	No Load		Full Load	
Mem. (MB)	CPU (%)	Mem. (MB)	CPU (%)	
Van.	603.877	4.024	603.877	5.436
TC	623.313	4.028	624.084	5.759
Incr.	**3.22%**	**0.09%**	**3.35%**	**5.94%**

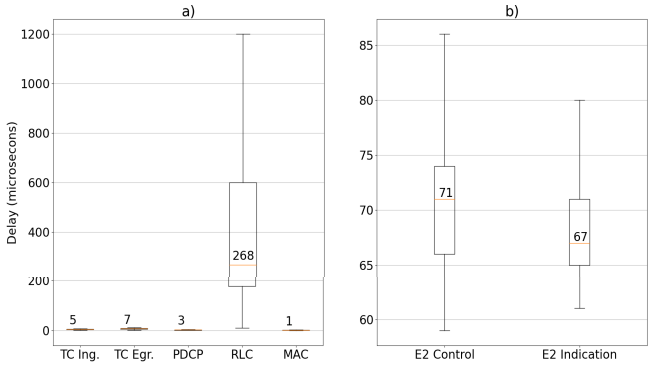

Fig. 4. Disaggregated latency per sublayer in downlink and xApp E2 Control and E2 Indication communication latencies.

The classifier, queue, scheduler and pacer stages are designed as shared objects with a common interface, similar to how loadable kernel modules are designed (i.e., at start the module function pointers are initialized[6]), and thus, they can be reconfigured online, enabling fine-grained control of the proposed TC RF pipeline from an xApp.

We did not use recursive mechanisms in TC-RAN. However, in our design, recursive structures within shared objects can be easily implemented if needed.

V. EVALUATION

In this section, we first describe our testbed and the overhead of TC RF in comparison with vanilla OAI [54]. Following, we validate TC-RAN's per-flow traffic slicing and low-latency capabilities. Next we show a real online multiplayer game scenario, and we end this section with some results that aim to shed some light in the still open question of the adequacy of handling 5G QoS in the RAN against CCA solutions in a time-varying MCS cellular scenario. The code used in this manuscript for the RAN can be found in the official OAI's repository,[7] while nearRT-RIC's and xApp code is available at FlexRIC's official repository.[8]

A. Testbed and Performance Overhead

We prototyped TC RF on top of the OAI 5G SA stack.[9] Likewise, we connected our PC (i.e., Intel (R) Xeon (R) Gold

[6]See net/ipv4/tcp_bbr.c and struct tcp_congestion_ops definition in the linux kernel source code for an example.

[7]https://gitlab.eurecom.fr/oai/openairinterface5g/-/tree/tc-ran

[8]https://gitlab.eurecom.fr/mosaic5g/flexric/-/tree/master/

[9]Commit 50d9c1f2bfaf094e60178f830cad4c8ddc3039d6.

6208U @ 2.90 GHz with 16 cores) to a USRP B210 capable of achieving a 40 MHz bandwidth. For creating our TC SM we chose FlexRIC due to its easy integration with OAI, as well as its lean, clear and extensible architecture to create SMs *Ã la carte* in comparison with the nearRT-RIC provided by O-RAN or the Open Network Foundation (ONF). Furthermore, we utilized the Quectel 5G RM510Q-GL, connected to another PC (i.e., Intel (R) Core (TM) i7-8550U CPU @ 1.80GHz), as well as a Huawei P40 phone as UEs. We used the 5G band n78 (i.e, TDD), subcarrier spacing μ 30kHz, 1 component carrier, 1 layer and 106 PRB, with a TTI of 500 μs and a configuration of 7/2/1 D/U/M. For the tested scenarios, we used *iperf3* for emulating a greedy flow and the ping utility tool to emulate a time-sensitive application, as well as to measure the RTT. Unless explicitly mentioned, we used TCP Cubic. For measuring the extra resources consumed by our TC RF implementation, we measured two scenarios: (i) no load where only the ping utility is running and, (ii) full load where 4 greedy flows in parallel are instantiated. As observed in Table II, the TC RF built on top of OAI consumes 3.22% and 3.35% more memory and 0.09% and 5.94% more normalized CPU than vanilla OAI, for no load and full load cases, respectively. Therefore, the cost of adding the TC RF is small and relatively negligible with other CPU and memory demanding tasks. The results here shown represent an average of 10 different runs.

We also analyzed the contribution to the delay budget per sublayer in an unloaded OAI RAN, as well as the delay between the xApp and the gNodeB for E2 Control and E2 Indication messages, both of them generated with a periodicity of 1 ms. As observed in Fig. 4a, the total downlink latency of a packet is dominated by the RLC sublayer. This is due to the fact that we are using a TDD configuration, so whenever the packet arrives into the RLC sublayer, it has to synchronize with a downlink transmission opportunity which in the worst case may occur 1.5 ms later (i.e., in the DDDDDDDMUU pattern the packet could arrive just a moment after the M has been scheduled, and thus, it needs to wait the full M and the following two UU uplinks, each of which consume 500 μs). Regarding TC RF implementation, the packet processing latency is close to 25 times less than the one introduced by RLC, and thus, it does not play any significant role in the total delay of the packet. Regarding the xApp to gNode communication delay, in our testbed, where the xApp, nearRT-RIC and E2 Agent embedded into the gNodeB run in the same machine on different processes, the measured median latencies are 71 and 67 μs for the E2 Control (xApp to E2 Agent) and E2 Indication messages (from E2 Agent to xApp), respectively. These values are also more than two orders of magnitude lower than the required O-RAN value of 10 ms for nearRT-RIC communications.

B. Validation

In this subsection, we validate the TC-RAN's capabilities and its versatility. For stability purposes we limited the MCS to 10.

1) RAN Flow Slicing: Similar to how the MAC Resource Block Groups (RBGs) can be sliced (e.g., assigning a percentage of RBGs to a specific RLC buffer), so can be achieved

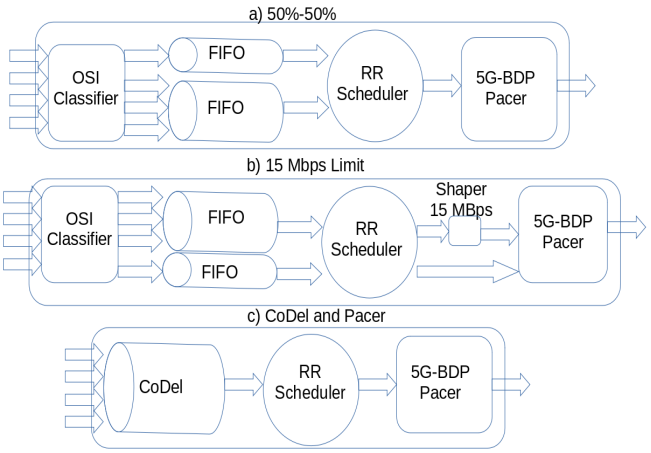

Fig. 5. 50%-50%, 15 MBps and CoDel TC RF deployed stages in the tests.

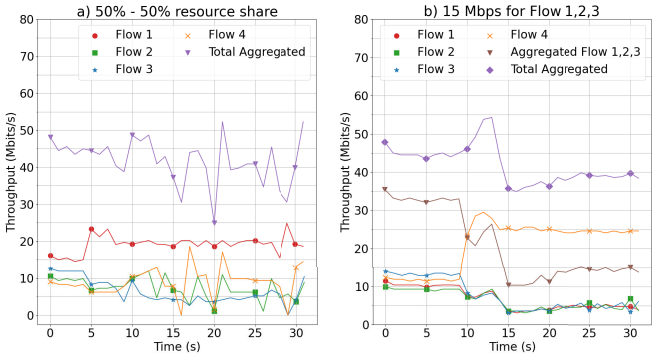

Fig. 6. 50 - 50 % resource share between Flow 1 and Flow 2,3,4 and 15 Mbps shaper limit for Flow 1,2,3.

with data flows. In Fig. 6a, four flows in downlink can be observed. Around the 5th second, the TC RF is reconfigured through an xApp with an OSI classifier, a second FIFO queue, a RR scheduler and a 5G-BDP pacer, as shown in Fig. 5a. The OSI classifier will segregate the flows according to their 5-tuple when queuing, while the RR scheduler will fairly dequeue the packets from the available queues. Additionally, a control message is sent to the classifier, aiming to segregate the Flows 2, 3, and 4 into an independent queue, emulating an application that consists of 3 flows, while the rest of the flows (i.e., Flow 1) are routed to the default queue. As a result the resources are equally distributed among the 2 queues with a 50%-50% percentages, due to the scheduler's RR algorithm, achieving both a 20 Mbps rate, as observed in Fig. 6a. Note that without the pacer, the packets would have ingress in the RLC directly, similar to the vanilla case, and no slicing effects would have been perceived. Other requirements of flow slices may involve limiting the traffic of a slice to a data rate, similar to how the QoS in 5G is specified [3]. In Fig. 6b, the effect of deploying a similar TC RF configuration (i.e., Fig. 5b) around the 10th second can be observed, but in this scenario a shaper, limiting the rate to 15 Mbps is also deployed. In this case, an xApp aggregates the Flow 1,2, and 3 into the first queue as observed in Fig. 5b leading to a 15 Mbps aggregated throughput in contrast to the 25 Mbps that achieve the rest of the flows that are dispatched to the second queue (i.e., Flow 4). Note that 3GPP lacks a mechanism to achieve per-flow slicing for flows marked with the same QFI as the two described

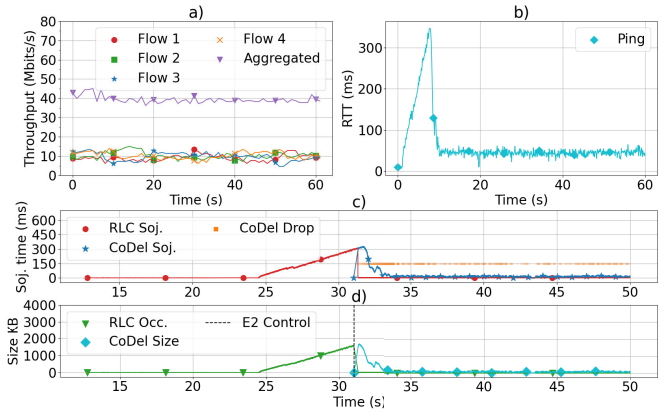

Fig. 7. CoDel and 5G-BDP effect in the throughput, latency and RLC buffer.

scenarios. The bandwidth of a flow can be limited from the UPF, but there is no mechanism to partition the resources among the flows marked with the same QFI, in a system where the bandwidth can fluctuate notably (e.g., due to channel conditions). Hence, the results shown here are not achievable in contemporary standard 3GPP cellular networks.

2) Low Latency: For applications with stringent latency requirements, 3GPP guarantees a maximum delay a packet shall experience [3]. However, it does not define the means to achieve it. In Fig. 7, four flows in downlink along with a time-sensitive flow are instantiated. From Fig. 7d, it can be observed how the TCP flows start accumulating in the RLC sublayer buffer, causing the RLC buffer to grow beyond 1.5 MBs. A linear relation between the RLC buffer size and the sojourn time from Fig.7c of a packet in the RLC buffer can also be observed. Besides, in this scenario, the delay perceived by the time-sensitive flow is mostly governed by the delay generated at the RLC queue, going above 300 ms and thus, breaking any low-latency guarantee, as shown in Fig. 7b. To remedy it, we developed an xApp with a control loop that triggers the TC RF configuration shown in Fig. 5d. The pipeline deployed by the xApp includes a CoDel queue with default values (i.e., interval/target 100/5 ms), a RR Scheduler, and a 5G-BDP pacer, as shown in Fig. 5c. It can be observed from the Fig. 7c and 7d that at the 31st second (Fig. 7a and Fig. 7b's time are not synchronized with Fig. 7c and Fig. 7d), the RLC buffer occupancy starts decreasing, while the CoDel queue starts increasing. The sojourn time that occurs in the RLC queue is maintained for some time, due to the amount of packets that were already enqueued, but starts decreasing afterwards. Note here the effectiveness of the 5G-BDP pacer in maintaining the RLC buffer with enough data not to starve the MAC scheduler, while not bloating it, as no throughput decrease is observed. Fig. 7c also shows the moments where, due to excessive sojourn time, CoDel drops packets. The Y axis for these events is omitted and are only plotted in the case where a packet is dropped. At the first moments where CoDel is deployed (i.e., around the 31st second), the pace at which the CCA (i.e., Cubic) sends packets is considerably higher than the channel throughput. Therefore, CoDel drops packets more frequently at the beginning, reducing the queue size, while still maintaining enough packets to feed the MAC scheduler. 3GPP foresees different behaviors for different QFIs. Therefore, this scenario

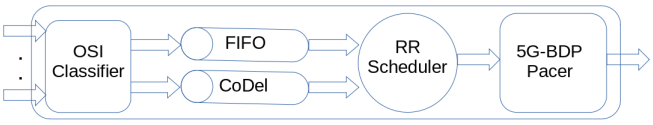

Fig. 8. Deployed TC RF pipeline while playing slither.io.

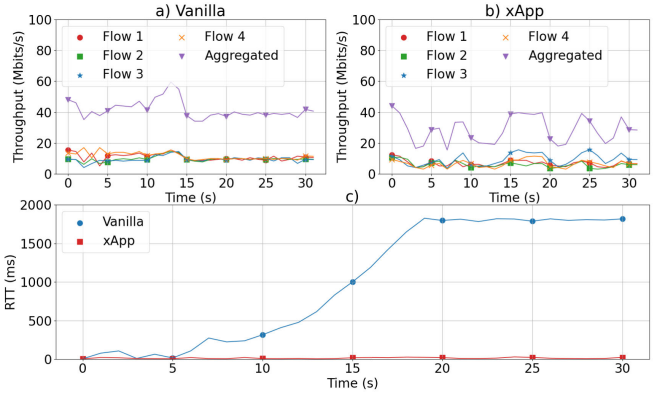

Fig. 9. Network perspective: throughput and latency while playing slither.io.

could be avoided in contemporary cellular networks if the time sensitive packet could be segregated with a different QFI and enqueued in a different buffer from the rest of the flows. However, this may not always be possible (e.g., the content provider does not inform the service provider). Additionally, 3GPP does not expect any AQM mechanism, and thus, it does not exist a standard manner to communicate to the sender to reduce its pace (e.g., using ECN or dropping a packet) before bloating the buffer. Moreover, 3GPP's model foresees a communication with the Core Network to mark the packets with different QFIs, which may lay geographically distant, contrary to the proposed solution, that is instantiated locally.

C. Online Multiplayer Game Scenario: Slither.io

Lastly, we demonstrate the flexibility and effectiveness of the proposed TC-RAN system with the Slither[10] online multiplayer game played on a Huawei P40 5G Android phone, in conjunction with a greedy background traffic using Android's termux terminal emulator, for which we provide a video.[11] In this manner, the results can be analyzed from three different perspectives: (i) end users QoE playing the online game, (ii) network performance through *ping* and *iperf3*, and (iii) RAN's perspective through RLC sublayer's statistics. The xApp subscribes to the RLC SM measuring the sojourn time through the E2 Indication message. When it detects that the sojourn time of the packets trespass 10 ms, it deploys a second CoDel queue and deviates the greedy flows to them, as shown in Fig. 8. In this manner, the QoE perceived by the user is maintained. From the video, in the vanilla case, it can be noticed that the game becomes unresponsive 5 seconds after the background traffic starts, ruining the user experience. This is completely different from the second case, where the background flows are segregated into a second CoDel queue. The video results confirm that with the deployed TC

[10]http://slither.io/
[11]https://vimeo.com/783683553

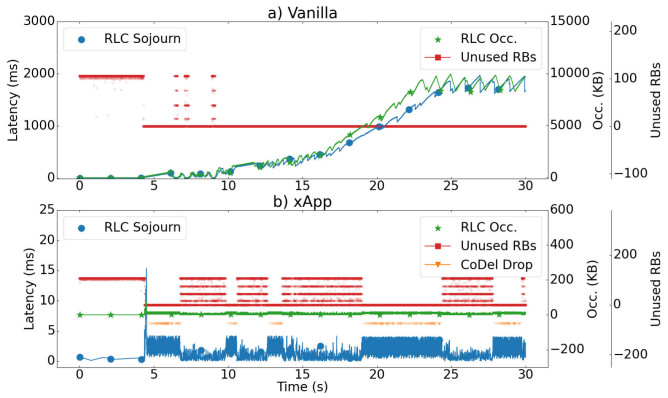

Fig. 10. RAN perspective: RLC buffer occupancy, sojourn time, unused RBs and packets drops while playing slither.io.

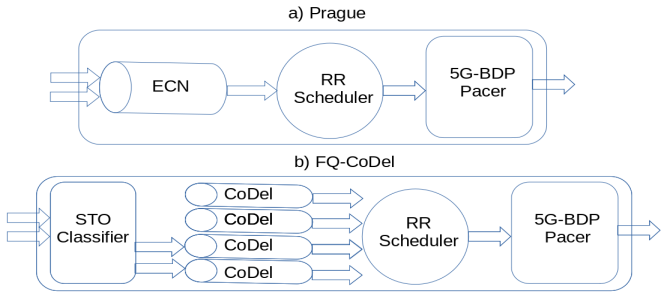

Fig. 11. TC RF deployed stages.

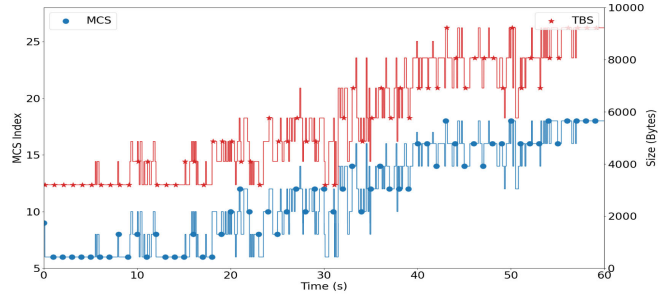

Fig. 12. MCS dynamic profile and the corresponding TBS.

RF configuration, the user experience significantly improves, as the game continues smoothly while the greedy traffic does not cease flowing in the background. Meanwhile, Fig. 9 shows the network perspective. In the vanilla case, the achieved throughput arrives to 40 Mbps while in the xApp case, 10 Mbps are not utilized and only 30 Mbps are reported. However, the lost bandwidth is compensated by the latency, as segregating the traffic maintains the time sensitive traffic delay around 20 ms while in the vanilla case, the delay increases up to 1.8 seconds, as illustrated by Fig. 9c. As observed in Fig. 10, from the RAN's perspective, in the vanilla case, after 10 seconds the background traffic has generated a buffer of 1 MBytes, and it continues increasing the buffer, until it completely bloats the buffer with 10 MBytes that cause delays in the RLC buffer of around 1.8 seconds, completely ruining the user experience. Fig. 10a also shows that during the seconds 5-10, there are TTIs where there is not enough data to fulfil the MAC scheduler capacity. Therefore, in the video, the game is still responsive during the first 5 seconds after the background traffic has started. On the other hand, when the TC RF configuration from Fig. 8 is deployed through the xApp, the latency in the RLC buffer never surpasses 20 ms, and thus, the game can be played normally, at the cost of not fully using all the available RBs. Fig. 10 also shows CoDel's instants where it drops packets, which coincide with the moments where all the RBs are utilized, due to the fact that a queue is forming and CoDel tries to maintain it unbloated. This dichotomy of bandwidth vs delay needs to be interpreted through an xApp that is capable of assigning semantic to the network behavior and act accordingly. As an example, in this scenario, sacrificing bandwidth in favor of reducing the delay seems the most sensible solution, as it enables a smooth QoE from a user's perspective at the cost of a small amount of bandwidth. However, this decision is arbitrary and context dependent, and thus, it is xApp's responsibility to decipher the context, assign a semantic and deploy the appropriate configuration to achieve the required SLAs. This real world scenario shows 3GPP's weakness to achieve the goals for which it was designed. Some applications (e.g., OTT applications) will not trigger a new DRB creation, and thus, they will end sharing the RLC buffer. Therefore, in such cases segregating the traffic marked with the same QFI is crucial if some fairness per-flow is required. In this scenario, E2 TC also shows its versatility, as queues from different types can be instantiated, and different traffic flows can be segregated belonging to the same QFI in near real-time.

To sum up, TC-RAN's composability, customizability and configurability have been evaluated. TC-RAN permits smoothly composing different pipelines tailored to diverse use cases, configuring the stages as needed and configuring the pipeline online, provisioning means to achieve the demanded 5G QoS requirements that vanilla 3GPP lacks.

D. Case Study: Low-Latency and High Throughput in Cellular Networks

Even though it is known from a theoretical point of view that no distributed CCA can achieve low-latency and high throughput [19], new CCAs flourished in recent years trying to work as close as possible to the optimal point [5], [39]. On the other hand, other solutions try to tackle the problem before the slowest link in the path (e.g., FQ-CoDel), as they can gather key information to optimize the pacing rate and reduce the sojourn time. Either way, the question of the best location remains open with new proposed solutions continuously emerging [32], [55], [56].

Therefore, and trying to shed some light, in this subsection, we benchmark two CCAs in 5G that aim not to bloat the buffers while maintaining some per-flow fairness and high throughput, namely BBRv2 and Prague [39], against a traditional solution (i.e., FQ-CoDel) deployed through TC-RAN in a dynamic MCS scenario, as shown in Fig. 11. For testing Prague, we enabled ECN in both endpoints and marked the packet if the sojourn time was above 5 ms during a 100 ms interval, similar to CoDel. This deployment is similar to the L4S deployment in 5G [41], but we measure the sojourn time at the TC RF queue rather than the sojourn time at the RLC sublayer, and we only test it with "scalable" flows (i.e., TCP Prague). This deployment mimics L4S architecture and is possible due to the versatile 6 stage architecture of TC-RAN.

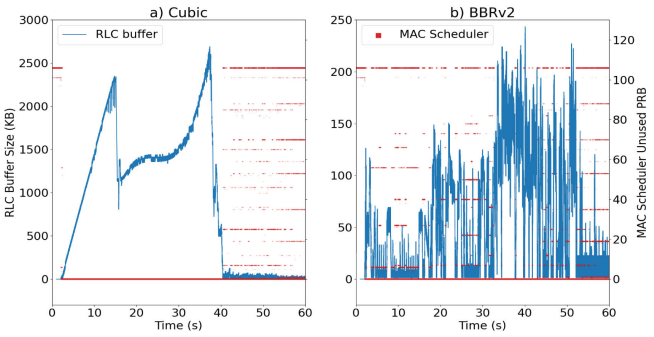

Fig. 13. RLC buffer status and RB utilization in a dynamic MCS scenario for TCP Cubic and TCP BBRv2.

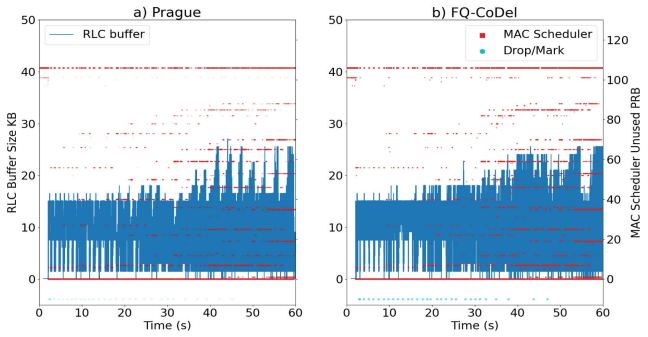

Fig. 14. RLC buffer status and RB utilization in a dynamic MCS scenario for TCP Prague and FQ-CoDel with TCP Cubic.

Fig. 12 illustrates the MCS Index variation in the range $[6, 18]$ for a duration of 60 seconds, representing user channel quality improvement over time. Note that the throughput is tightly correlated with the MCS, as shown in Fig. 12. In fact, the optimal point to work, in a channel with zero losses, shall only have enough bytes to fulfil the TBS of the current TTI. Any additional byte adds delay, while any missing byte results in lost transmission opportunities. Fig. 13 and Fig. 14 show the RLC buffer status of the four considered scenarios. As expected, Cubic uses all the RBs available from the RAN during the first 40 seconds, at the cost of accumulating up to 2.5 MBytes in the RLC buffer. During the last 20 seconds, the MCS increases, and so does the link capacity, achieving more than 9000 bytes per TTI, as shown in Fig. 12. However, Cubic is not capable of correctly estimating the new channel capacity, and thus, there exists lost transmission opportunities where no data is sent. BBRv2, on the other hand, is able to correctly estimate the channel capacity increase that occurs in the last 20 seconds, and thus, it forwards more bytes, trying to always maintain the RLC buffer with enough bytes not to starve the MAC scheduler. However, when analyzing RB utilization, BBRv2's channel capacity estimation leads to non-negligible unused RBs during the whole experiment, while in the case of Cubic, only during the last 20 seconds RBs are not utilized. In the case of Prague, packets are marked with the ECN bit, to inform the sender that congestion is happening, which mostly occurs in the first part of the experiment, as the channel capacity is modest, while no packet marking occurs in the last 20 seconds when the capacity increases, as shown in Fig. 14a. Note that Prague pipeline requires a pacer (i.e., 5G-BDP in this experiment), to control the size of the RLC buffers. It can also be observed that Prague also misses a non-negligible amount of transmission opportunities, as illustrated in Fig. 14a, especially when the channel quality increases. Lastly, we deployed a more traditional system segregating the flows (i.e., greedy and time-sensitive) with the well-known CoDel queues using the TC-RAN system through an xApp. As observed, in Fig. 14b, the behavior is similar to Prague but with a better RB utilization. More interesting conclusions can be drawn from Table III and Fig. 15. In the Table III , the mean throughput reported by *iperf3* is shown. Surprisingly, BBRv2 is the method that reaches the highest throughput (i.e., 44.60) followed by Cubic (i.e., 42.30), FQ-CoDel (i.e., 41.72), and Prague (i.e., 39.58). The fact that Cubic uses all the available RBs during the first part of the experiment, where the MCS is low, and thus, so is the available throughput, does not compensate the lost transmission opportunities that occur later. Regarding the latency, Cubic's bloated RLC buffer causes up to 500 ms delays, while BBRv2, Prague and FQ-CoDel, maintain a lower delay, close to 20 ms each. Moreover, interesting conclusions can be reached from the Delay vs used RBs graph in Fig. 15a , where the optimal point for the system to work is the latency obtained when there is no load in the system (i.e., 10 ms in this deployment) and where all the RBs are used. Cubic uses most of the RBs (i.e., 86%) while the mean delay is 210 ms, far from many time-sensitive use cases in 5G. FQ-CoDel achieves a more appealing result using 82% of the available RBs, while suffers a 21 ms mean delay, while BBRv2 utilizes 83% of all the available RBs with a mean 30 ms delay and Prague shows also competitive 78% RB utilization and 20 ms mean delay.

TABLE III
MEAN THROUGHPUT

	Cubic	BBRv2	Prague	FQ-CoDel
Throughput (Mbps)	42.30	44.60	39.58	41.72

This scenario validates a static FQ-CoDel design with 5G specificities (i.e., 5G-BDP pacer) against CCA approaches and shows current 3GPP QoS model limitations as only one queue per QFI is foreseen even with non-loss based CCA. Besides, this subsection shows that addressing the problem near the location where it occurs (i.e., the RLC buffer) results in better yet comparable results over solutions that tackle the problem through the CCA (i.e., BBRv2 and Prague) and act accordingly. Furthermore, it also shows the shortages of current CCAs, where there exists a mechanism to inform them to slow down (i.e., dropping/marking a packet) while there is no way to notify them that starvation is happening, and thus, RBs are squandered. Therefore, mechanisms as the one proposed at [57], where starvation can be reported to the CCA to accelerate its forwarding rate, may be promising solutions for cellular networks, where low-latency guarantees can be achieved, but not full bandwidth.

TC-RAN's versatility to unleash 5G QoS means is shown in this section. Current Linux default TCP CCA's (i.e., Cubic) behavior ruins most of the time-sensitive use cases described by 3GPP, and it is not capable of dynamically adjusting its rate to the cellular network channel variability. However, TC-RAN, through its composable, customizable and reconfigurable flexible design, provides the means to deploy different configurations to achieve stringent time and resource demands.

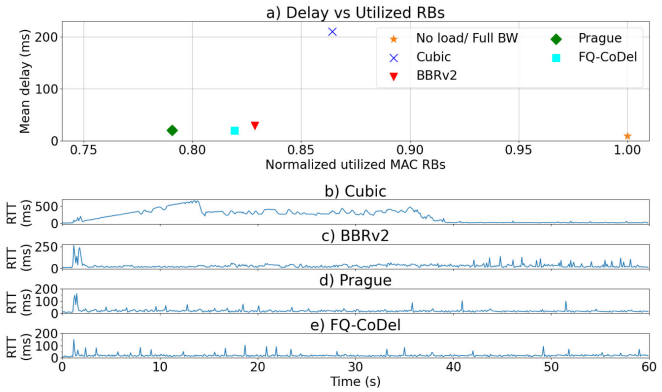

Fig. 15. Mean Delay vs Normalized Utilized PRBs and RTT of Cubic, BBRv2, Prague and FQ-CoDel.

Using TC-RAN, xApps are empowered to dynamically compose, customize and reconfigure the data flow pipelines in near real-time considering the RAN status, and thus, optimize the overall performance. Additionally, some limitations when trying to optimize the QoS from the CCA have been shown.

VI. Conclusion

In this paper, we present TC-RAN, an O-RAN compatible E2SM for promoting traffic flows to first-class citizens in cellular networks, modifying the current 3GPP stack, while embracing the Open RAN paradigm. As demonstrated in this paper, there exist concrete and realistic scenarios that neither 3GPP nor O-RAN address, that are of utmost importance for guaranteeing the QoS in cellular networks. The presented TC SM is a fundamental pillar to embrace the 5G/6G QoS requirements and manage the traffic flows through a standard interface (i.e., E2SM) in the RAN, that opens a new field for cross-optimization algorithms in AI/ML to combine RB allocation with per-flow QoS. The presented 6 stage pipeline segregates the responsibility of each module, following the principle of *Do One Thing and Do It Well* and enabling extension, customization, composition and programmability. In this manner, TC-RAN provides the means to compose complex architectures seamlessly (e.g., FQ-CoDel), customize them tailoring to specific scenarios (e.g., multiplayer online game), extend them with new components (e.g., new AQM algorithms), and reconfigure them as required. Additionally, results shown in Section V-C highlight the QoE improvement that a UE can notice in a multiplayer online game scenario using TC-RAN, avoiding large response times of over a second when diverse flows share the DRB. Moreover, Section V-D demonstrates that addressing the QoS from the CCA in a dynamic channel scenario is very challenging, and that CCAs, that provide low-latency services are distant from the optimal working point (i.e., Prague 22% squandered RBs and x2 higher latency than the no-loaded network and BBRv2 18% squandered RBs and x3 higher latency in the tested scenario). Therefore, addressing the problem close to the slowest data path link is of upmost importance, for which TC-RAN provides the means. Furthermore, it shows the necessity of a communication link between the RAN and the CCA to avoid squandering RBs.

TC-RAN is open source[12] [13] under a permissive license (i.e., OAI's FRAND license)[14] to promote its adoption and use. As future work, we plan to extend the TC-RAN with more pluggable entities (e.g., RED AQM queue), devise new abstractions to facilitate xApp control loop (e.g, SLA on total throughput) and port the TC RF into other cellular network stacks (e.g., srsRAN).

References

[1] (2022). *O-Ran Alliance*. [Online]. Available: https://www.o-ran.org/
[2] T. Høiland-Jørgensen, C. A. Grazia, P. Hurtig, and A. Brunstrom, "Flent: The flexible network tester," in *Proc. 11th EAI Int. Conf. Perform. Eval. Methodologies Tools*. New York, NY, USA: Association for Computing Machinery, Dec. 2017, pp. 120–125.
[3] *System Architecture for the 5G System*, document TR 23.501, version 18.1.0, 3GPP, 2023.
[4] *Voip Bandwidth Consume*. [Online]. Available: https://www.cisco.com/c/en/us/support/docs/voice/voice-quality/7934-bwidth-consume.html
[5] 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.
[6] P. Yang, X. Xi, T. Q. S. Quek, J. Chen, X. Cao, and D. Wu, "How should I orchestrate resources of my slices for bursty URLLC service provision?" *IEEE Trans. Commun.*, vol. 69, no. 2, pp. 1134–1146, Feb. 2021.
[7] P. Yang, X. Xi, T. Q. S. Quek, J. Chen, X. Cao, and D. Wu, "RAN slicing for massive IoT and bursty URLLC service multiplexing: Analysis and optimization," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 14258–14275, Sep. 2021.
[8] *O-RAN E2 Service Model (E2SM), RAN Control*, O-RAN Alliance, Alfter, Germany, document O-RAN.WG3.E2SM-RC-v01.03, Version 1.03, Oct. 2022.
[9] *NR, Radio Resource Control (RRC) Protocol Specification*, document TS 38.331, Version 17.0.0, 3GPP, Apr. 2022.
[10] 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 *Proc. IEEE 29th Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2018, pp. 1–6.
[11] T. Høiland-Jørgensen, M. Kazior, D. Taht, P. Hurtig, and A. Brunstrom, "Ending the anomaly: Achieving low latency and airtime fairness in WiFi," in *Proc. USENIX Annu. Tech. Conf.*, Santa Clara, CA, USA, 2017, pp. 139–151.
[12] (2022). *The Bufferbloat Project*. [Online]. Available: https://www.bufferbloat.net/projects/
[13] B. Briscoe, K. D. Schepper, M. B. Braun, and G. White. (Mar. 2020). *Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service: Internet Engineering Task Force*. [Online]. Available: https://tools.ietf.org/html/draft-ietf-tsvwg-l4s-arch-06
[14] *Evolved Universal Terrestrial Radio Access (E-UTRA) and NR; Service Data Adaptation Protocol (SDAP) Specification*, document TR 37.324, Version 17.0.0, 3GPP, 2022.
[15] M. Irazabal, E. Lopez-Aguilera, I. Demirkol, R. Schmidt, and N. Nikaein, "Preventing RLC buffer sojourn delays in 5G," *IEEE Access*, vol. 9, pp. 39466–39488, 2021.
[16] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, pp. 64–74, Jul. 2008.
[17] L. Kleinrock, "Internet congestion control using the power metric: Keep the pipe just full, but no fuller," *Ad Hoc Netw.*, vol. 80, pp. 142–157, Nov. 2018.
[18] L. Kleinrock, "Power and deterministic rules of thumb for probabilistic problems in computer communications," in *Proc. Conf. Rec., Int. Conf. Commun.*, Boston, MA, USA, 1979, p. 43.
[19] J. Jaffe, "Flow control power is nondecentralizable," *IEEE Trans. Commun.*, vol. COM-29, no. 9, pp. 1301–1306, Sep. 1981.
[20] B. Kernighan and D. Ritchie, *The C Programming Language*, vol. 1. Upper Saddle River, NJ, USA: Prentice-Hall, 1978.

[12] https://gitlab.eurecom.fr/oai/openairinterface5g/-/tree/tc-ran
[13] https://gitlab.eurecom.fr/mosaic5g/flexric/-/tree/master/
[14] https://openairinterface.org/legal/oai-public-license/

[21] N. McKeown et al., "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[22] P. Bosshart et al., "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014.

[23] R. MacDavid et al., "A P4-based 5G user plane function," in *Proc. ACM SIGCOMM Symp. SDN Res.* New York, NY, USA: Association for Computing Machinery, 2021, pp. 162–168.

[24] *O-RAN E2 Service Model (E2SM), RAN Function Network Interface*, O-RAN Alliance, Alfter, Germany, document ORAN-WG3.E2SM-NI-v01.00, version 1.0, Feb. 2020.

[25] *O-RAN E2 Service Model (E2SM), Key Performance Monitoring*, O-RAN Alliance, Alfter, Germany, document O-RAN.WG3.E2SM-KPM-v02.03, Version 2.03, Oct. 2022.

[26] *O-RAN E2 Service Model (E2SM), Cell Configuration and Control*, O-RAN Alliance, Alfter, Germany, document O-RAN.WG3.E2SM-CCC-v01.00, Version 1.03, Oct. 2022.

[27] R. Schmidt, M. Irazabal, and N. Nikaein, "FlexRIC: An SDK for next-Generation SD-RANs," in *Proc. 17th Int. Conf. Emerg. Netw. Exp. Technol.* New York, NY, USA: Association for Computing Machinery, 2021, pp. 411–425.

[28] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "ColO-RAN: Developing machine learning-based xApps for open RAN closed-loop control on programmable experimental platforms," *IEEE Trans. Mobile Comput.*, vol. 22, no. 10, pp. 5787–5800, Oct. 2023.

[29] S. D'Oro, M. Polese, L. Bonati, H. Cheng, and T. Melodia, "DApps: Distributed applications for real-time inference and control in O-RAN," *IEEE Commun. Mag.*, vol. 60, no. 11, pp. 52–58, Nov. 2022.

[30] H. Haile, K.-J. Grinnemo, S. Ferlin, P. Hurtig, and A. Brunstrom, "End-to-end congestion control approaches for high throughput and low delay in 4G/5G cellular networks," *Comput. Netw.*, vol. 186, Feb. 2021, Art. no. 107692.

[31] S. Alfredsson, G. Del Giudice, J. Garcia, A. Brunstrom, L. De Cicco, and S. Mascolo, "Impact of TCP congestion control on bufferbloat in cellular networks," in *Proc. IEEE 14th Int. Symp. World Wireless, Mobile Multimedia Networks (WoWMoM)*, Jun. 2013, pp. 1–7.

[32] S. Abbasloo, C.-Y. Yen, and H. J. Chao, "Wanna make your TCP scheme great for cellular networks? Let machines do it for you!" *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 265–279, Jan. 2021.

[33] P. Imputato, S. Avallone, M. P. Tahiliani, and G. Ramakrishnan, "Revisiting design choices in queue disciplines: The PIE case," *Comput. Netw.*, vol. 171, Apr. 2020, Art. no. 107136.

[34] (2022). *Cake*. [Online]. Available: https://github.com/sqm-autorate/sqm-autorate

[35] (2022). *10th Anniversary of Codel*. [Online]. Available: https://lists.bufferbloat.net/pipermail/cerowrt-devel/2012-May/000233.html

[36] J. Corbet. (2012). *TCP Small Queues*. [Online]. Available: https://lwn.net/Articles/506237/

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

[38] C. Pupiales, D. Laselva, and I. Demirkol, "Capacity and congestion aware flow control mechanism for efficient traffic aggregation in multi-radio dual connectivity," *IEEE Access*, vol. 9, pp. 114929–114944, 2021.

[39] (2022). *TCP Prague*. [Online]. Available: https://github.com/L4STeam/tcp-prague

[40] (2022). *Enabling Time-Critical Applications Over 5G With Rate Adaptation*. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/white-papers/enabling-time-critical-applications-over-5g-with-rate-adaptation

[41] D. Brunello, I. Johansson S, M. Ozger, and C. Cavdar, "Low latency low loss scalable throughput in 5G networks," in *Proc. IEEE 93rd Veh. Technol. Conf. (VTC-Spring)*, Apr. 2021, pp. 1–7.

[42] Q. Liu, N. Choi, and T. Han, "OnSlicing: Online end-to-end network slicing with reinforcement learning," in *Proc. 17th Int. Conf. Emerg. Netw. EXperiments Technol.* New York, NY, USA: Association for Computing Machinery, Dec. 2021, pp. 141–153, doi: 10.1145/3485983.3494850.

[43] Q. Liu, N. Choi, and T. Han, "Deep reinforcement learning for end-to-end network slicing: Challenges and solutions," *IEEE Netw.*, vol. 37, no. 2, pp. 222–228, Mar./Apr. 2023.

[44] *NR; Packet Data Convergence Protocol (PDCP) Specification*, document TS 38.323, Version 17.0.0, 3GPP, 2022.

[45] *NR User Plane Protocol*, document TS 38.425, Version 16.1.0, 3GPP, 2020.

[46] P. E. McKenney, "Stochastic fairness queueing," in *Proc. IEEE Int. Conf. Comput. Commun.*, 1990, pp. 733–740.

[47] M. Irazabal, E. Lopez-Aguilera, I. Demirkol, and N. Nikaein, "Dynamic buffer sizing and pacing as enablers of 5G low-latency services," *IEEE Trans. Mobile Comput.*, vol. 21, no. 3, pp. 926–939, Mar. 2022.

[48] J. Gettys, "Bufferbloat: Dark buffers in the internet," *IEEE Internet Comput.*, vol. 15, no. 3, p. 96, May 2011.

[49] P. Popovski et al., "Wireless access for ultra-reliable low-latency communication: Principles and building blocks," *IEEE Netw.*, vol. 32, no. 2, pp. 16–23, Mar. 2018.

[50] *NG Application Protocol (NGAP)*, document TS 38.413, Version 15.0.0, 3GPP, 2018.

[51] M. D. McIlroy, E. N. Pinson, and B. A. Tague, "UNIX time-sharing system: Foreword," *Bell Syst. Tech. J.*, vol. 57, no. 6, pp. 1899–1904, Jul. 1978.

[52] K. Nichols and V. Jacobson, "Controlling queue delay," *Queue*, vol. 10, no. 5, pp. 20–34, May 2012.

[53] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug. 1993.

[54] *Open Air Interface*. Accessed: May 16, 2023. [Online]. Available: https://www.openairinterface.org/

[55] S. Abbasloo, Y. Xu, and H. J. Chao, "C2TCP: A flexible cellular TCP to meet stringent delay requirements," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 4, pp. 918–932, Apr. 2019.

[56] J. Zhou et al., "A machine learning-based framework for dynamic selection of congestion control algorithms," *IEEE/ACM Trans. Netw.*, vol. 31, no. 4, pp. 1566–1581, Aug. 2023.

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

Mikel Irazabal received the M.Sc. and Ph.D. degrees from Universitat Politècnica de Catalunya (UPC).

He was an Early Stage Researcher (ESR) at the European-funded Innovative Training Network (ITN), Marie Skłodowska-Curie Actions (MSCA). He is currently a Technology Consultant, who offers high-quality 5G software development services. He is also an Architect and a main Maintainer at FlexRIC, where he has developed critical performance modules for the OpenAirInterface Software Alliance, such as a scalable thread pool. His research interests include low-latency and programmable wireless communication systems.

Navid Nikaein is currently a Professor with the Department of Communication Systems, Eurecom. At Eurecom, he is leading the Research and Development Group on experimental 4G/5G system research for emerging use cases found in private 5G and Open RAN. He is also the Founder and the CEO of BubbleRAN. At BubbleRAN, he is helping organizations to seamlessly build, operate, and automate their private 4G/5G network infrastructure by consolidating open RAN architecture and cloud-native intelligence to offer greater efficiency, scalability, and performance. He is the Founder of Mosaic5G, the Co-Founder of OpenAirInterface, and a Board Member of OpenAirInterface Software Alliance.