Active Queue Management in Disaggregated 5G and Beyond Cellular Networks using Machine Learning

Alexandros Stoltidis and Kostas Choumas and Thanasis Korakis
Dept. of ECE, University of Thessaly, Volos, Greece
Email: stalexandros, kohoumas, korakis@uth.gr

***Abstract*—In the context of 5G and beyond cellular networks, this paper delves into Active Queue Management (AQM) implementation in high-latency environments within disaggregated Radio Access Network (RAN) deployments, addressing bufferbloat while improving overall end-to-end network performance. While researchers investigate AQM algorithms to mitigate bufferbloat in monolithic RAN deployments, they overlook the inherent capabilities of disaggregation in 5G and beyond cellular networks, which involves intricate cross-layer communication among distinct network entities housing different protocol stack layers. Our study explores the 5G architecture, investigates previous research on AQM, identifies challenges arising from these algorithms in disaggregated network configurations, and proposes a comprehensive scheme for managing AQM in these configurations. To facilitate our approach, we leverage RAN Intelligent Controller (RIC) entities equipped with Artificial Intelligence (AI) and Machine Learning (ML). We evaluated our novel solution by implementing a previously proposed AQM algorithm, known as Dynamic RLC Queue Limit (DRQL), in a disaggregated RAN deployment within a high-latency network environment and assessing its effectiveness through the Quality of Service (QoS) achieved at the NITOS testbed, utilizing OpenAirInterface5G (OAI5G).**

***Index Terms*—5G, QoS, AQM, AI, ML, LSTM, Disaggregated RAN, RIC, OpenAirInterface5G.**

I. Introduction

The fifth-generation (5G) technology promises to revolutionize how we communicate and interact with the world around us. Unlike its predecessors, 5G is not just about faster speeds but also about providing a platform for a range of innovative services. To achieve this, 5G has introduced three service categories [1], called Enhanced Mobile Broadband (eMBB), Ultra-Reliable Low-Latency Communications (URLLC), and Massive Machine-Type Communications (mMTC). eMBB supports high-speed data applications and services, URLLC focuses on applications that demand ultra-reliable and low-latency communication, and mMTC enables communication between loads of devices while consuming little energy and transmitting at low data rates.

Modern networks employ large buffers for efficient resource utilization, affecting these services with the bufferbloat phenomenon [2]. As network buffers become excessively large, low-latency-sensitive flows face prolonged delays and unavoidable sojourn times within these buffers, leading to high latency and consecutively degraded network performance. High latency reduces the available bandwidth for eMBB, undermines the low-latency requirements of URLLC, and causes network congestion and increased network resource consumption in mMTC. Thus, bufferbloat presents a noteworthy issue in modern networks, resulting in delays, congestion, and suboptimal overall performance. AQM is a technique used to prevent bufferbloat by actively managing the size of the buffers in network devices and entities. It aims to replace traditional passive queue management techniques, such as drop-tail and drop-head, to manage increasing network congestion by maintaining buffers at a reasonable size, preventing them from becoming extensively large and causing congestion or delays. The literature has introduced several AQM algorithms to manage queues across different protocol stack layers within the 5G architecture. However, many of these algorithms primarily target 5G's monolithic deployment, often neglecting its disaggregated characteristics. Regarding disaggregation, the network protocol stack layers, including their functions and corresponding queues, are split between separate network entities. As a result, these algorithms may prove inadequate in disaggregated network deployments.

As the 5G specification does not address AQM in disaggregated deployments, it is crucial to integrate it within the 5G framework or consider it for future generations of cellular networks. Consequently, our research focuses on formulating a comprehensive approach to manage AQM algorithms within 5G and beyond networks. We leverage AI/ML within each RIC to tackle challenges in disaggregated network deployments. To showcase the effectiveness of using AI/ML in such environments, we evaluate an AQM algorithm where the information exchanged between the 5G network entities is forecasted. Subsequently, we evaluate the performance of this AQM algorithm in monolithic and disaggregated 5G networks, utilizing either exchanged or forecasted information, to measure the degree of improvement achieved.

This paper comprises five primary sections. Section I provides a comprehensive introduction to the bufferbloat phenomenon and AQM. Section II delves into the existing literature on AQM in 5G networks. In Sections III and IV, the proposed implementation is described and assessed through an evaluation framework. We conclude this paper in Section V while discussing future research directions of AQM beyond 5G networks.

II. Related Work

All AQM algorithms operate on the fundamental principle of minimizing the number of packets in the queues by discarding packets without data starving the transmission channel.

ISBN: 978-3-903176-61-4

© 2024 International Federation for Information Processing (IFIP).
ISBN: 978-3-903176-61-4

Research has shown that maintaining small queue sizes can significantly reduce sojourn times, resulting in lower latency [3]. The literature contains a wide range of AQM algorithms that operate per-queue independently of any communication between multiple 5G layers that may reside in separate network entities, thus eliminating the need for coordinated decision-making. Such examples are Random Early Detection (RED) [4], Controlled Delay (CoDel) [5], Fair Queuing CoDel (FQ-CoDel) [6] and Proportional Integral Controller Enhanced (PIE) [7]. The application of RED in cellular networks, which probabilistically discards packets at the RLC network stack layer, has been evaluated in [8]. Meanwhile, [3] presents the adoption of CoDel in cellular networks, where packets are dropped based on their sojourn time, reducing the overall latency. These algorithms function in a queue-agnostic manner, handling each queue independently without requiring additional information.

While algorithms like these are adequate, they may not be optimal, as they do not consider the network as a whole. Addressing this limitation requires the development of more sophisticated algorithms that need cross-layer communication for coordinated decision-making. Examples of such algorithms include Stochastic Fair Queuing (SFQ) [9], 5G Bandwidth Delay Product (5G-BDP), UPF-SDAP Pacer (USP), and DRQL [10]. Rather than relying on dropping packets, these algorithms limit transmission buffer sizes to manage network congestion. In [11], the successful implementation of SFQ in LTE networks, also known as Dynamic RLC Queue Management, has been reported and demonstrated promising results. In SFQ, communication between the RLC and PDCP layers is necessary. 5G-BDP, on the other hand, necessitates communication among the MAC, RLC, and SDAP layers. In this regard, the MAC layer interfaces with the RLC layer, while the SDAP layer interfaces with the MAC layer. Moreover, USP mandates communication between the SDAP and RLC layers and the User Plane Function (UPF) in the 5G Core Network (5G-CN). UPF communicates with the SDAP layer, and in turn, the SDAP layer communicates with the RLC layer.

Finally, DRQL relies on the communication between the SDAP and RLC layers, as it operates by having the SDAP layer continuously querying the RLC layer for its buffer's maximum allowed capacity and occupancy. This information helps determine whether to forward packets or not. As the RLC buffers fill up, the MAC layer extracts as much data as the radio channel can support. When the MAC pulls a full RLC buffer, it becomes starved, indicating that it can handle more data and leading to an increase in the RLC buffer's maximum allowed capacity. However, if the MAC layer extracts only a portion of the data from the RLC buffer, leaving some data in the buffer indicates the necessity to reduce the maximum allowed capacity to match the radio channel capacity. While this paper concentrates on the DRQL algorithm due to its simplicity, the proposed scheme can apply to any AQM algorithm that necessitates cross-layer communication between layers located in separate network entities.

III. AQM in Disaggregated Networks

A. 5G Architecture and Proposed AQM Solution

Although there is rich literature on AQM in 5G and beyond networks, it mostly assumes a monolithic 5G-RAN. In the context of 5G cellular networks, a base station previously known as Node-B (NB) is now called Next-Generation NB (gNB), which distinguishes it from the Enhanced NB (eNB) used in 4G networks. Contrary to the monolithic eNB's design, the gNB's design focuses on virtualization and cloud-native capabilities. The optimization of gNB's functionality and the cloudification of the network is the result of the functional splits introduced by 3GPP, allowing the gNB to disaggregate into multiple entities. These entities are the Centralized Unit (CU) and the Distributed Unit (DU). The most widely adopted functional split option is the 3GPP 7.2x split [12], according to which CU supports the higher layers of the protocol stack encompassing SDAP, PDCP, and RRC, Meanwhile, DU supports the lower layers, including RLC, MAC and PHY. The CU comprises the control plane (CU-CP), which includes the RRC layer, and the user plane (CU-UP), which contains the SDAP layer, as more thoroughly described in [13]. Figure 1 also illustrates the protocol stack layers in control and user planes in monolithic and disaggregated RAN deployments, using a green and a red vertical line, respectively.

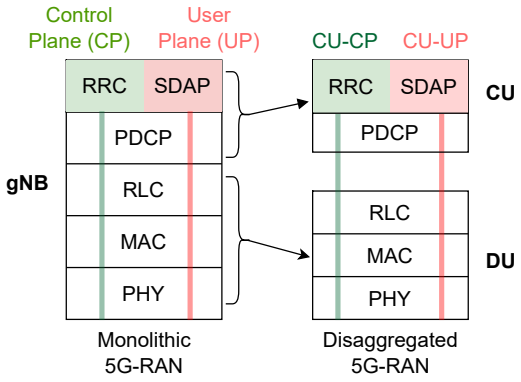

Fig. 1: Control and user planes in monolithic and disaggregated gNB deployment.

The integration of most AQM algorithms into 5G-RAN requires some minor architectural modifications. Despite the typical 5G-RAN architecture only allowing for RLC queues at DU entities, the exploitation of several AQM algorithms requires the existence of SDAP queues at CU entities, as Figure 2 depicts. Each CU has a set of SDAP queues mapped to different QoS flows, each identified by a unique QoS Flow Identifier (QFI). Each downlink packet is tagged with its appropriate QFI by UPF, according to its Packet Detection Rule (PDR), and then inserted into the respective SDAP queues at CU. The SDAP scheduler is in charge of forwarding these packets to the appropriate Data Radio Bearer (DRB) at the RLC layer of DU. Furthermore, as we have already mentioned, some AQM algorithms rely on cross-layer communication,

ISBN: 978-3-903176-61-4

which takes place between protocol stack layers situated on different CU and DU entities. In DRQL, for instance, communication between SDAP at CU and RLC at DU is necessary. In the case of a monolithic gNB, CU and DU entities coexist on a single host machine, facilitating almost instantaneous communication between them. In contrast, a disaggregated gNB might separate CU and DU entities onto different network nodes, unavoidably introducing communication delay between them. Hence, a disaggregated gNB can significantly impact the performance of latency-sensitive AQM algorithms.

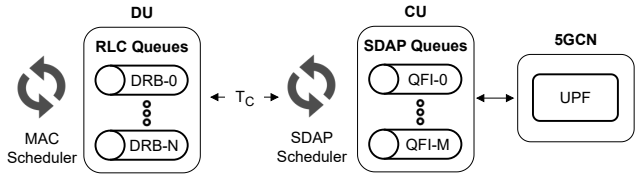

Fig. 2: RLC and SDAP queues in dissagregated gNB.

Our proposed solution for AQM utilizes AI/ML to address the challenges posed by the communication delay in disaggregated gNB deployments. We adopt the Software-Defined Networking (SDN) approach and O-RAN specification, according to which 5G-RAN decouples part of the control plane and moves to the RIC entities that exploit AI/ML. To avoid the delay introduced by the communication between CU and DU, the RIC entities can predict the state of DU for decision-making at CU rather than relying on their direct, high-latency information exchange. To be more precise, RIC entities can monitor the behavior and status of the DU, such as fluctuations in its queues, and make predictions about its future state by analyzing historical patterns. Then, CU leverages the forecasts produced to make informed decisions on packet forwarding. To thoroughly assess the effectiveness of our approach, we conducted a comprehensive evaluation of the DRQL AQM algorithm in a disaggregated RAN deployment, which was previously evaluated exclusively in the context of a monolithic gNB, as documented in [10].

B. RIC Assisted DRQL

As with most AQM algorithms, DRQL establishes seamless cross-layer communication between CU and DU. For successful downlink traffic transmission from CU to DU, DRQL necessitates precise measurements of RLC queue status. These measurements provide information about the actual sizes and limits of the RLC queues, enabling efficient packet pacing and resource allocation. The efficient DRQL operation in a disaggregated gNB requires time-critical delivery of the RLC queue measurements from DU to the SDAP layer at CU. Although CU can request this information from DU using its control interface, due to communication delay, the information provided refers to a previous RLC queue status, hindering the real-time decision-making at CU. Denoting the symmetrical and bi-directional communication delay between CU and DU as T_c, the information that CU requested at $t_0 - 2T_c$ and received at t_0 was generated by DU at $t_0 - T_c$. Consequently, the SDAP scheduler might make suboptimal decisions regarding packet forwarding, relying on outdated information. Furthermore, since the SDAP requires this information for every available packet residing in its queues to facilitate packet forwarding, the delay associated with acquiring this information contributes to an overall latency increase while concurrently diminishing throughput.

In our approach, instead of SDAP repeatedly querying measurements of the RLC queue status whenever there is a packet to be forwarded, we employ RIC to predict these measurements, analyzing historical data. Periodically, DU informs the two separate RIC entities, named Non-Real Time RIC (Non-RT RIC) and Near-Real Time RIC (Near-RT RIC), of its current RLC queue measurements. It's worth noting that 5G network control loops determine the use of different RIC entities. There are near-real time loops, carried out by CU entities and the Near-RT RIC, last between 10ms and 1s, while non-real time loops, executed by the Non-RT RIC, last longer than 1s[1].

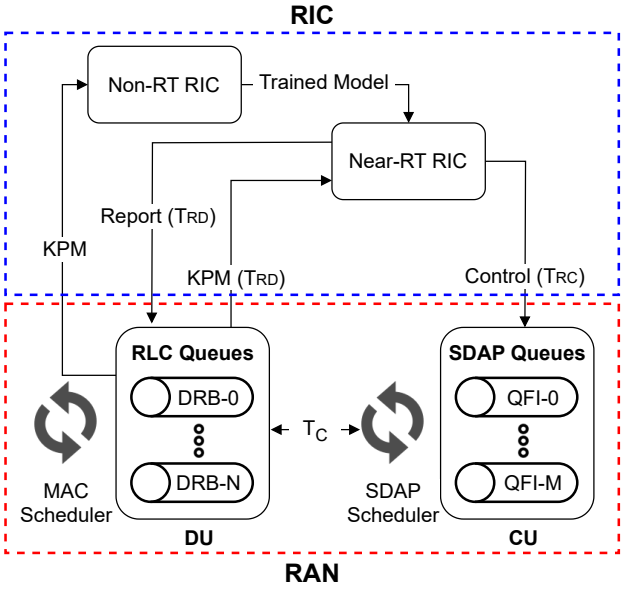

Fig. 3: Communication between RIC entities and the CU and DU.

The Non-RT RIC trains the AI/ML models using the collected measurements and transfers them into the Near-RT RIC. Its data collection and, therefore, training processes are performed in a monolithic network deployment, ensuring the specified, proper DRQL operation and valid monitoring of the anticipated fluctuations in the RLC queues whenever the MAC scheduler extracts data, every Transmission Time Interval (TTI). The Near-RT RIC utilizes the trained AI/ML models to make informed control decisions through its microservices, named xApps, based on the current received measurements,

[1]These two loop types are Loop 2 and 3, respectively, since Loop 1 is the real-time control loop performed by DUs and RUs and operates on timescales of 1ms.

ISBN: 978-3-903176-61-4

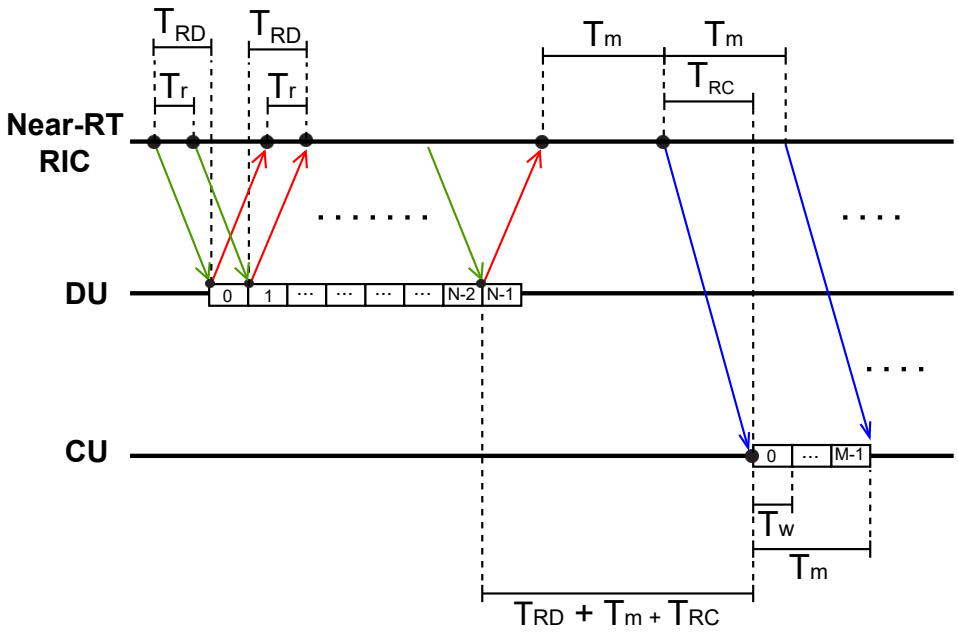

Fig. 4: Timing of Near-RT RIC, CU and DU.

as Figure 3 depicts. To collect the RLC queue measurements, Near-RT RIC continuously requests E2SM-KPM [14] using report messages received from the DU. The Near-RT RIC forecasts the RLC queue fluctuations and notifies the CU of the anticipated RLC queue status. The CU can use the data received from the Near-RT RIC to avoid the T_c latency.

C. Crucial Factors for In-Time Forecasting

The communication latencies between the Near-RT RIC and the CU or the DU, named T_{RC} and T_{RD} respectively, must be considered for efficient model training and inference. As the left part of Figure 4 illustrates, the Near-RT RIC periodically sends report messages to the DU every T_r to obtain the RLC queue measurements. These report messages reach the DU with a T_{RD} latency, and when a report request is received, the DU responds to the Near-RT RIC with the requested information. Therefore, the latter receives these responses periodically at intervals of T_r, with each response also incurring the T_{RD} latency.

As illustrated in the right part of Figure 4, the Near-RT RIC periodically uses N responses from the DU, collected over a duration NT_r, to predict M values related to the RLC queue status, representing a duration of MT_w. The inference process duration of the model, named T_m, is factored in when making predictions, ensuring accurate RLC queue predictions at the precise moment. The model also considers the previously mentioned latencies, including T_{RD} and T_{RC}. Therefore, the Near-RT RIC incorporates the T_{RD}, T_{RC}, and T_m latencies in its predictions, which delivers to CU after a cumulative elapsed time of $T_{RD} + T_m + T_{RC}$ from the time reference at the DU, corresponding to the most recently received RLC measurement. The CU receives these predictions repeatedly with a period of T_m.

Finally, since the Non-RT RIC conducts model training on datasets extracted in real-time timescales, the trained models can exclusively perceive and forecast the fluctuations in the RLC queues that transpire within these timeframes. However, the SDAP scheduler diverges from the received predictions by not restricting packet transmissions to these timescales but instead forwards packets as soon as they become available at the SDAP queues. As a result, for a point in the $T_w = T_m/M$ interval between two consecutive predictions given by Near-RT RIC, the RLC queue status must be estimated by CU on its own. Since the RLC status information comprises numerical values, our proposed implementation considers a linear variability of the RLC queue status across multiple packets within the T_w time window and provides predictions with their corresponding confidence intervals. Given the received values x_n and x_{n+1} representing the RLC status at time points t_n and $t_{n+1} = t_n + T_w$, the CU can perform linear interpolation to estimate the RLC status, x, for forwarding a packet at t within the t_n and t_{n+1} interval. This estimation is achieved through the formula $x = x_n + \frac{x_{n+1}-x_n}{t_{n+1}-t_n}(t - t_n)$, where x is the estimated RLC status at time $t \in [t_n, t_{n+1}]$.

IV. Evaluation Framework

This section dissects the evaluation parameters, the 5G configuration, and the measurement methodologies employed to simulate and assess our proposed solution. It's worth highlighting that the simplicity of our experimental setup is a result of constraints in hardware and computational resources. Furthermore, it's essential to emphasize that the research's primary focus is not solely on the ML approach but rather serves as a proof of concept, with the potential for expansion to accommodate more intricate environments.

ISBN: 978-3-903176-61-4

A. 5G Configuration

Our implementation, named OAI5G-DRQL, is an extension of the original OAI5G [15], a 5G-RAN implementation, where we introduced several enhancements to the SDAP layer, including multiple SDAP queues and a Round-Robin (RR) scheduler for packet forwarding. We have also modified the RLC layer, enabling it to dynamically adjust the limits of its queues based on the incoming packet volume. Moreover, we extended the currently implemented KPM sent from DU to CU to accommodate additional RLC queue-specific information, as expected in E2SM-KPM. The Radio Frequency (RF) Simulator, provided by OAI5G, simulates the physical layer of 5G-RAN and allows testing without an RF board, while NR-UE emulates a commercial off-the-shelf (COTS) User Equipment (UE) connected to 5G-RAN. A `Python3` Non-RT RIC performs data preprocessing and model training, while FlexRIC [16] is the Near-RT RIC as it connects to the 5G-RAN nodes via the E2 interface, allowing monitoring and control via report and control messages. Additionally, we deploy the 5G-CN using OAI5G Core Network [17]. To accurately configure delays between network entities, we deploy the CU, DU, UE, Near-RT RIC, and Non-RT RIC on a shared computational NITOS node, utilizing traffic control (`tc`). Finally, a separate node is hosting the 5G-CN.

B. Dataset Creation

In the context of predicting the behavior of RLC queues in a supervised learning manner, creating a comprehensive dataset under a range of network conditions is imperative. The developed dataset should encompass bufferbloat scenarios and provide insights into the behavior of network entities when operating in the presence of an AQM algorithm. In our case, we employ `iperf3` [18] and `ping` [19] to incorporate bufferbloat into our dataset by creating **simultaneous high-throughput and latency-sensitive downlink traffic with varying throughput and interval parameters for encapsulating different network application requirements**. By adjusting the downlink transmission rate in `iperf3`, we replicate diverse network applications, including file downloading, video streaming, and cloud gaming. Additionally, varying the `ping` interval allows us to emulate different network requirements associated with applications like online gaming and video conferencing.

More specifically, the developed dataset contains the RLC's queue limits and actual sizes, sampled at T_r = 1 ms under DRQL in a monolithic network deployment, characterized by $T_c = 0$. Employing combinations of different parameters in `iperf3` and `ping` creates distinct network downlink conditions. These combinations include UDP traffic at $[1, 10, 40, 100]$ Mbps alongside ICMP transmission with intervals of $[1, 10, 100]$ ms, ensuring the accuracy and relevance of the trained models' predictions across diverse applications. Therefore, the compiled dataset includes entries representing the RLC's queue limit and actual size, along with the respective timestamps of these observations.

C. ML for Queue Fluctuation Predictions

Our preliminary studies and experiments indicate that Long Short-Term Memory (LSTM) models outperform Seasonal Autoregressive Integrated Moving Average (SARIMA) and Random Forests for time-series forecasting of RLC's queue fluctuations in DRQL. Thus, Non-RT RIC uses LSTM to develop and train a pair of time series forecasting models to predict the forthcoming fluctuations of the RLC's queue limits and actual sizes, respectively. Additionally, the Non-RT RIC deploys both models to the Near-RT RIC, facilitating decision-making through forecasting.

The Non-RT RIC initiates model training by initially preprocessing the provided dataset. In the preprocessing stage, the Non-RT RIC scales the data in the $[0, 1]$ range using `MinMaxScaler`, provided by scikit-learn [20], and splits the dataset into 70% training, 20% validation, and 10% test sets. The feature-label pairs are derived using a rolling window approach, preserving causality by refraining from data shuffling. The input features consist of N data points, representing a time duration of NT_r, while the output labels comprise M data points, equivalent to a duration of $MT_w = T_m$, as previously noted. Therefore, for a given set of features N, the labels become M following a time interval of $T_{RD} + T_m + T_{RC}$ to account for DU to Near-RT RIC, model inference, and Near-RT RIC to CU delays. Finally, to ensure consistency and simplicity in our feature-label alignment, we define $T_w = T_r$.

The Non-RT RIC employs TensorFlow [21] with Keras [22] to handle the development and training of the LSTM models, while KerasTuner optimizes training hyperparameters for precise forecasting results. We employ `tensorflow.keras.sequential` for both models to stack multiple layers. We specify that the input layer for the two models is an LSTM layer consisting of N nodes. KerasTuner determines the parameters for the subsequent layers, including their quantity and configurations. Additionally, we add a dropout layer, in the range of 0 to 0.5 and a step size of 0.1, that helps prevent overfitting by randomly deactivating a fraction of neurons during training. The final layer is a dense layer, which is fully connected and produces predictions aligned with the dimensions of our target labels, denoted as M in this context. We utilize the Mean Squared Error (MSE) as our chosen loss function, effectively penalizing errors and providing remarkable overall model performance. Finally, the choice of optimizer algorithm is pivotal in shaping how errors propagate through the network. The Adam optimizer is an excellent choice since it strikes an optimal balance between learning the most relevant and less frequent features.

It's important to emphasize that accurately forecasting the precise limits and actual sizes of RLC's queues is complex, necessitating supplementary data from all CU and DU entities. However, in our straightforward proof-of-concept implementation, minor discrepancies in our predictions only lead to marginal reductions in throughput or increases in latency due to minimal packet drops within a single Transmission Time Interval (TTI). Furthermore, determining the model's

ISBN: 978-3-903176-61-4

inference duration T_m is a formidable task, and therefore, we calculate it as the mean inference duration obtained from multiple observations of model inference. Finally, the significance of available hardware and network infrastructure, influencing the inference duration and latency between CU and DU, along with their corresponding Near-RT RIC, respectively, becomes evident in prediction accuracy since lower inference duration and reduced latency create a shorter rolling window for predictions. Given additional computing power, we could facilitate parallel model inference, thereby reducing the forecasting horizon and enhancing result accuracy. Nevertheless, within this uncomplicated implementation, such enhancement remains unnecessary.

D. Conducted Experiments

The experiments investigate the impact of coexisting high-throughput and latency-sensitive traffic on overall network performance, particularly in achieved throughput and latency, within a 5G network environment. Downlink UDP traffic is generated from 5G-CN at a rate of 50 Mbps, while at the same time, latency-sensitive ICMP packets are also transmitted downlink to the UE at 100 ms intervals. We measure the **throughput** of the UDP traffic and the **Round Trip Time (RTT)** of the ICMP traffic, which escalates as the packet's sojourn time increases. As illustrated in Figure 5, the UE establishes a single Packet Data Unit (PDU) session with distinct QFIs for UDP or ICMP packets, and a single RLC Acknowledge Mode DRB is responsible for transmitting and receiving both QoS flows.

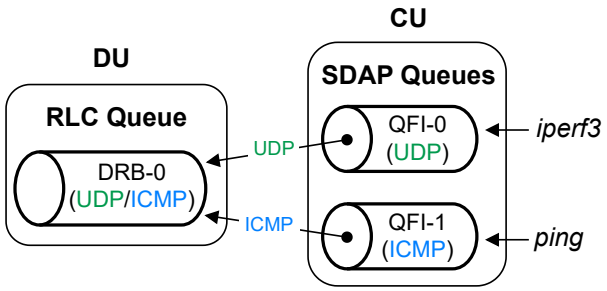

Fig. 5: ICMP and UDP SDAP QFI Queues Mapped to a Single RLC DRB.

We conduct **three sets of experiments** in either a **disaggregated** or **monolithic** 5G-RAN deployment, distinguished by $T_c \approx 10$ ms and $T_c \approx 0$, respectively. **In the first set, we use the original OAI5G** implementation, where SDAP forwards packets as soon as they become available to the RLC layer without querying RLC's status. It's important to note that the original OAI5G implementation utilizes a drop-tail approach for AQM in the RLC queues. This approach restricts the queue size and discards incoming packets that surpass this limit. We adjust the RLC queue limit to mitigate the drop-tail approach and increase queue occupancy to assess network performance in scenarios where the queues are congested, and therefore bufferbloat is more likely to occur. **In the second set of experiments, we use our OAI5G-DRQL without RIC** to eliminate the direct CU-DU communication. Finally, **in the third set of experiments, we evaluate OAI5G-DRQL**, where the Near-RT RIC utilizes the trained LSTM models to provide the CU with the anticipated RLC status in advance, and the DU is not involved. The Near-RT RIC is collocated with the CU, thus the delay between Near-RT RIC and DU is $T_{RD} = T_c \approx 10$ ms, while the delay between Near-RT RIC and CU is $T_{RC} \approx 0$.

When Near-RT RIC is used, the two LSTM models analyze $N = 1000$ data points of RLC queue statistics, which correspond to a duration of $1000T_r = 1000$ ms. Each model forecasts $M = 400$ values, representing the RLC queues status during the next $400T_w = 400$ ms, from when the CU receives these forecasts. The Near-RT RIC generates these predictions accounting for the model inference time $T_m \approx 400$ ms and the reporting latencies $T_{RC} \approx 0$ and $T_{RD} \approx 10$ ms, using 1000 ms of historical data to predict for the next 400 ms after a time interval of $T_{RD} + T_m + T_{RC} \approx 410$ ms, starting at the generation of the latest DU KPM used.

Through these experiments, we demonstrate the performance contrast between the original OAI5G without AQM and the proposed OAI5G-DRQL but without RIC, when confronted with bufferbloat in monolithic and disaggregated deployment scenarios. We also evaluate our RIC assisted OAI5G-DRQL, striving to mitigate the issues that surface in OAI5G-DRQL without RIC within high latency disaggregated deployments and achieve performance parity with the latter, implemented in non-latency monolithic deployments.

E. Experiment Results

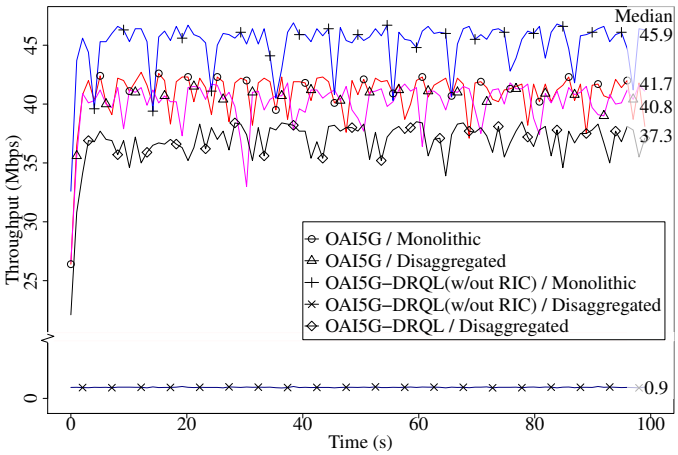

Fig. 6: Maximum achieved downlink throughput.

1) Throughput results: Figure 6 shows a graphic representation of the throughput achieved for each set of experiments. It is evident from the results that each approach exhibits a relatively low initial throughput, which gradually stabilizes and eventually achieves a higher throughput level. The observed low initial throughput is because the RLC queues are empty at the beginning of the experiments. As the queues fill

ISBN: 978-3-903176-61-4

up, the MAC scheduler can pull more data from the RLC queues, resulting in higher throughput. The inherent capacity limitations of the wireless channel are a ceiling that stabilizes the maximum achievable throughput.

In the first set of experiments, the CU in the original OAI5G maintains a consistent packet forwarding rate. Thus, throughput remains largely unaffected by the monolithic or disaggregated deployment, represented by a red circle-pointed line or a pink triangle-pointed line, respectively, averaging between the median of these two lines at around 41.2 Mbps.

In the second set of experiments, the OAI5G-DRQL without RIC outperforms the original OAI5G in a monolithic deployment, succeeding a throughput almost equal to 46 Mbps, illustrated as the median of the blue cross-pointed line. This increase in throughput can be attributed to the dynamic fluctuation of the RLC queues in response to the available radio channel capacity. These fluctuations ensure that the MAC scheduler processes precisely the amount of data required within a single TTI, avoiding RLC queue starvation and overflow. On the other hand, in a disaggregated deployment, OAI5G-DRQL without RIC features very low throughput at almost 1 Mbps, represented by the blue x-pointed line. This sharp throughput decrease arises from the fact that every packet reaching the CU undergoes the high-latency DRQL request-response process in SDAP, causing a bottleneck at CU, leading to RLC queue starvation and underutilization of the radio channel.

This is the rationale behind the usage of RIC in a disaggregated environment. As we see in the third set of experiments, OAI5G-DRQL with RIC effectively avoids the latency-heavy CU-DU communication, resulting in a throughput almost equal to 37 Mbps, which is the median of the black diamond-pointed line. This performance is slightly worse than the 41.2 Mbps throughput of the original OAI5G, but this is a tolerable price for the significantly higher performance of OAI5G-DRQL in terms of RTT, as it is presented below. We should also mention that the RIC is collocated with the DU in the same NITOS node, resulting in fewer available CPU resources for the MAC scheduler, which likely reduces the throughput of our scheme. In an actual non-experimental deployment with separate CPUs for each network entity, our implementation's throughput would be higher.

2) RTT results: The RTT is also examined in the context of high throughput to assess the effectiveness of our implementation. In Figure 7, we utilize a logarithmic scale to visualize the lower and higher RTT values throughout our experiments within the same graph.

In the original OAI5G of our first set of experiments, SDAP forwards packets as they become available without employing an SDAP scheduler, which causes a significant disparity in the volume of UDP packets compared to ICMP packets in the RLC queues and causes ICMP packets to lag significantly behind the UDP packets. As illustrated by the circular and triangular-pointed lines, this imbalance results in a linear increase in RTT over time, regardless of whether a communication delay exists between the CU and DU. Compared to

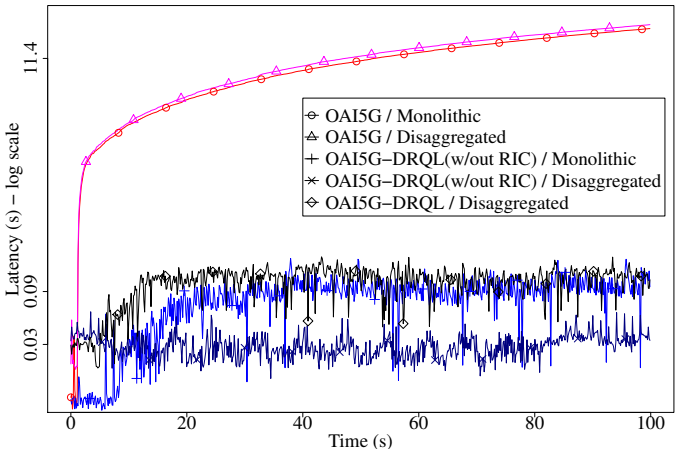

Fig. 7: Achieved RTT.

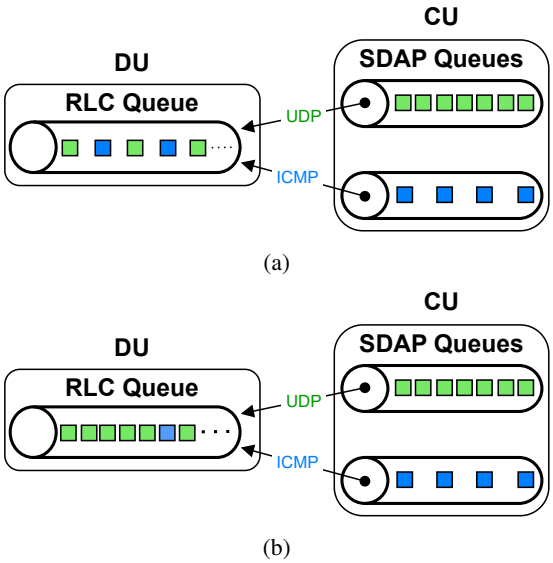

Fig. 8: Queues occupancy and RLC ICMP and UDP packet ratio and distribution in (a) disaggregated OAI5G-DRQL without RIC and in (b) every other implementation and deployment option.

the monolithic deployment, the rate of RTT growth slightly increases in the disaggregated deployment due to an increase in the initial acceleration of accumulation of UDP packets in the RLC queues.

In the second and third experiment sets, we compare OAI5G-DRQL without RIC in a monolithic and disaggregated deployment alongside OAI5G-DRQL with RIC in a disaggregated deployment. In both these cases, downlink traffic does not experience the additional DRQL communication overhead, avoiding SDAP scheduler impedance and ensuring robust decision-making on packet forwarding. The corresponding results are depicted with the cross-pointed and diamond-pointed lines, resulting in approximately 84 ms and 114 ms average RTT, respectively. The additional 20 ms of the average

ISBN: 978-3-903176-61-4

RTT in the disaggregated environment occurs due to the additional $T_c \approx 10$ ms it takes for an ICMP packet to travel from CU to DU and back. As counterintuitive as it seems, OAI5G-DRQL without RIC features decreased RTT, almost equal to 30 ms, depicted with the x-pointed line. Since each traffic flow possesses a unique QFI and the SDAP scheduler operates in a Round-Robin fashion, it systematically forwards UDP and ICMP traffic interchangeably. However, despite the higher arrival rate of UDP packets at SDAP, DRQL forwards them at a notably slower pace, relative to their volume, due to the additional T_c latency, resulting in fewer UDP packets ahead of ICMP packets in the RLC queue, and thus lower RTT for them, as illustrated in Figure 8a. It's worth noting that these lower RTT values come at the expense of significantly reduced throughput. On the other hand, Figure 8b shows the increased occupancy of UDP packets in front of ICMP packets in the RLC queue, resulting in slightly increased but justifiable latency in every other implementation and deployment option.

In summary, the RIC's assistance in OAI5G-DRQL enhances throughput compared to OAI5G-DRQL without RIC, approaching the monolithic deployment while maintaining relatively low RTT values. These improvements enable addressing bufferbloat with improved AQM in disaggregated high-latency deployments.

V. Conclusions & Future Work

This paper introduces and evaluates an innovative scheme that addresses AQM in disaggregated 5G and beyond cellular networks. During the evaluation process with the proposed OAI5G-DRQL, the traffic exhibited a stable and low RTT when compared to the linearly increasing RTT of the original OAI5G, at the cost of a minor throughput decrease of approximately 11.7% and 9.3%, in disaggragated and monolithic deployments respectively. Our solution allows for the adaptation of various AQM algorithms, initially designed for traditional monolithic deployments, to be utilized in disaggregated networks while maintaining stable latency and high throughput levels. We selected the DRQL AQM algorithm due to its simplicity; however, for a more comprehensive assessment of the proposed scheme's effectiveness in handling AQM in disaggregated networks, future work should include testing other algorithms like SFQ, 5G-BDP, and USP. Testing such algorithms would offer additional insights into the scheme's capacity to tackle AQM challenges in disaggregated networks. Moreover, to further validate the effectiveness of our proposed scheme, it should also be evaluated under real-life scenarios using traffic generated by multiple users. To explore more alternatives, we should also compare the accuracy of other AI/ML models that use different KPM from different protocol stack layers based on the AQM algorithm implemented and the network's requirements. Finally, we will examine how straightforward the development of AI/ML models can be in the previously mentioned scenarios and the scheme's ability to make accurate predictions in more complex environments.

Acknowledgment

The research leading to these results has received funding from the European Horizon 2020 Programme for research, technological development and demonstration under Grant Agreement Number No. 101008468 (H2020 SLICES-SC). The European Union and its agencies are not liable or otherwise responsible for the contents of this document; its content reflects the view of its authors only.

References

[1] P. Popovski, K. F. Trillingsgaard, O. Simeone, and G. Durisi. 5G Wireless Network Slicing for eMBB, URLLC, and mMTC: A Communication-Theoretic View. *IEEE Access*, 6:55765–55779, 2018.

[2] M. Irazabal, E. Lopez-Aguilera, I. Demirkol, R. Schmidt, and N. Nikaein. Preventing RLC Buffer Sojourn Delays in 5G. *IEEE Access*, 9:39466–39488, 2021.

[3] M. Irazabal, E. Lopez-Aguilera, and I. Demirkol. Active Queue Management as Quality of Service Enabler for 5G Networks. In *Proc. EuCNC*, 2019.

[4] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[5] K. M. Nichols, V. Jacobson, A. McGregor, and J. R. Iyengar. Controlled Delay Active Queue Management. *RFC*, 8289:1–25, 2018.

[6] T. Høiland-Jørgensen, P. E. McKenney, D. Täht, J. Gettys, and E. Dumazet. The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm. *RFC*, 8290:1–25, 2018.

[7] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg. PIE: A lightweight control scheme to address the bufferbloat problem. In *Proc. IEEE High Performance Switching and Routing (HPSR)*, 2013.

[8] A. K. Paul, H. Kawakami, A. Tachibana, and T. Hasegawa. An AQM based congestion control for eNB RLC in 4G/LTE network. In *Proc. IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2016.

[9] P.E. McKenney. Stochastic fairness queueing. In *Proc. IEEE INFOCOM*, 1990.

[10] M. Irazabal, E. Lopez-Aguilera, I. Demirkol, and N. Nikaein. Dynamic Buffer Sizing and Pacing as Enablers of 5G Low-Latency Services. *IEEE Transactions on Mobile Computing*, 21(3):926–939, 2022.

[11] R. Kumar, A. Francini, S. Panwar, and S. Sharma. Dynamic control of RLC buffer size for latency minimization in mobile RAN. In *Proc. IEEE WCNC*, 2018.

[12] N. Makris, C. Zarafetas, P. Basaras, T. Korakis, N. Nikaein, and L. Tassiulas. Cloud-Based Convergence of Heterogeneous RANs in 5G Disaggregated Architectures. In *Proc. IEEE ICC*, 2018.

[13] Interface between the Control Plane and the User Plane nodes (3GPP TS 29.244 version 16.5.0 Release 16).

[14] O-RAN Near-Real-time RAN Intelligent Controller E2 Service Model (E2SM) KPM 2.0. O-RAN Working Group 3, July 2021.

[15] Open Air Interface (OAI). https://openairinterface.org/.

[16] R. Schmidt, M. Irazabal, and N. Nikaein. FlexRIC: An SDK for next-Generation SD-RANs. In *Proc. CoNEXT*, 2021.

[17] Open Air Core Network 5G. https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed.

[18] iPerf. https://iperf.fr/.

[19] Ping. https://linux.die.net/man/8/ping.

[20] Fabian Pedregosa et al. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.

[21] Martin Ababi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, 2016.

[22] Keras. https://keras.io/.

ISBN: 978-3-903176-61-4