Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Electrical, Electronics and Communications Engineering
(36^{th}cycle)

Resource orchestration solutions for service provisioning in 5G/B5G networks: A computational and experimental approach

By

Somreeta Pramanik

******

Supervisor(s):
Prof. Dr. Carla Fabiana Chiasserini
Prof. Dr. Adlen Ksentini, Co-Supervisor

Doctoral Examination Committee:
Prof. Dr. Jérome Harri, Eurecom, France
Dr. Walter Cerroni, Università degli Studi di Bologna, Italy
Prof. Dr. Daniele Trinchero, Politecnico di Torino
Dr. Alessandro Nordio, CNR-IEIIT
Prof. Dr. Adlen Ksentini, EURECOM

Politecnico di Torino
2024

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Somreeta Pramanik
2024

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

I would like to dedicate this thesis to my loving parents, brother and husband

Acknowledgements

I would like to express my gratitude to all the people who shared with me even a small part of this 3 years journey. I want to explicitly extend my appreciation to the people without whom this work would not have been the same.

First of all, I have to thank my advisor Prof. Carla Fabiana Chiasserini for her invaluable continuous support in guiding me toward this result. I sincerely appreciate all the assistance, insights, and advice she shared with me during these years. Her dedication and professionalism have truly been an inspiration. I would like to extend my gratitude to Prof. Adlen Ksentini for his guidance and support and for assisting with my research visit to Eurecom.

I want to thank all the colleagues and friends that I met at Politecnico. Among them, a big thank you to Corrado, who initially helped me to settle down in the lab and also discuss work in detail, especially with the test-bed. My thanks also to the whole research group, past and present members.

Next, I want to thank my husband and brother who have been constantly encouraging and supporting me while being together as well as helping me to overcome professional difficulties. Finally, I want to thank my parents and family for their long-distance encouragement and support. I can not express how much I appreciate you always being there for me. Additionally, this work was supported partially by the European Commission through Grant No.101095890 (project PREDICT-6G) and partially by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on “Telecommunications of the Future” (PE0000001 - program “RESTART”).

PhD Activity

Journal publications

- Cost-efficient Slicing in Virtual Radio Access Networks, Pramanik, Somreeta; Ksentini, Adlen; Chiasserini, Carla Fabiana, in: *Computer Communications,* 2023
- Fair and Scalable Orchestration of Network and Compute Resources for Virtual Edge Services, Tripathi, Sharda; Puligheddu, Corrado; Pramanik, Somreeta; Garcia-Saavedra, Andres; Chiasserini, Carla Fabiana, in: *IEEE Transactions on Mobile Computing*, 2023
- Cost-efficient RAN Slicing for Service Provisioning in 5G/B5G, Pramanik, Somreeta; Ksentini, Adlen; Chiasserini, Carla Fabiana, in: *Computer Communications,* (under review), 2023

Conference publications

- Characterizing the Computational and Memory Requirements of Virtual RANs, Pramanik, Somreeta; Ksentini, Adlen; Chiasserini, Carla Fabiana, IEEE/IFIP WONS 2022
- VERA: Resource Orchestration for Virtualized Services at the Edge, Tripathi, Sharda; Puligheddu, Corrado; Pramanik, Somreeta; Garcia-Saavedra, Andres; Chiasserini, Carla Fabiana, IEEE International Conference on Communications (IEEE ICC 2022)

Attended classes and passed tests

Name	Hours
Advanced scientific programming in matlab	30
Connected Vehicles,	20
Adversarial training of neural networks	15
Optical Transport Networks	30
Complex networks and telecommunications, 2d edition: Towards 6G	30
Statistical data processing	20
Communication	5
Design Thinking, Processes and Methods	2
Intercultural & interpersonal management	8
Managing conflict: negotiation and communication	5
Navigating the hiring process: CV, tests, interview	2
Personal branding	1
Project management	5
Public speaking	5
Research integrity	5
Responsible research and innovation, the impact on social challenges	5
The new Internet Society: entering the black-box of digital innovations	6
Thinking out of the box	1
Time management	2
Responsible research and innovation, the impact on social challenges	5
Time management	2
IEEE/DEI Summer Ph.D. School on Machine Learning, Sustainable Edge Computing and Networking	30
5G: ENABLING TECHNOLOGIES, OPPORTUNITIES AND RESEARCH CHALLENGES AHEAD	30

External training activity

Name	Location	Hours
Eurecom	Biot	-
Italian networking workshop INW 2022	Courmayeur (AO)	-

Abstract

Next-generation mobile networks (5G/B5G) are tailored to enable vertical industries to provide a diverse range of virtualized services to their users. However, the growing complexity of mobile services poses challenges in meeting demanding performance requirements. Specifically, this becomes significant with varying service and traffic demands over time without resorting to costly overprovisioning. Besides, the trade-off between spectral efficiency and the consumption of data processing resources presents a complex scheduling dilemma. To elaborate, the scarcity of radio spectrum necessitates efficient spectrum sharing to meet various service level agreements (SLAs). Simultaneously, the limited computing resources at the network edge underscore the importance of allocating virtualized resources in a computationally aware manner. The thesis elaborately discusses the above-mentioned issues and simultaneously proposes solutions that mitigate the problem.

Initially, to gain practical insights and key inputs to design efficient resource management in virtualized Radio Access Networks (vRANs), we investigate and characterize the computational and memory requirements of vRANs by developing a srsRAN-based test-bed. Through experiments, we profile the consumption of computing and memory resources, and we assess the system performance. Additionally, we construct regression models to predict system behavior with varying numbers of connected users and diverse radio transmission settings. This involves developing a methodology and prediction models to aid in the design and optimization of virtual RANs.

A step further, leveraging our experimental findings, we formulate a cost-efficient radio resource management in 5G featuring network slicing. We maximize the profit of all slices simultaneously guaranteeing the target data rate and delay specified in the SLAs for the different traffic flows. The numerical results substantiate the effectiveness of the solution, demonstrating a 10-15% reduction in radio resource

consumption for cost-efficient resource slicing while also accomplishing performance isolation and meeting the data rate and delay specified in the SLAs of, respectively, eMBB and uRLLC slices

Furthermore, this work contributes to the study of distributed RAN orchestration and edge resource management using reinforcement learning. The approach enables decision-making logic to be co-located with the services it controls, facilitating local fine-grained and low-latency actions. A key development is the VERA framework, designed for managing resources at the edge. VERA specifically addresses the concurrent execution of user applications and network functions, recognizing the interconnected resource requirements of these services. Taking into account LTE vRAN and a video transcoder as services, our proposed framework, VERA, is demonstrated to consistently meet service Key Performance Indicators (KPIs) over 96% of the observation periods.

Contents

Chapter 1

Introduction

Mobile data transmission is experiencing perpetual growth. The 2021 Ericsson Mobility Report reveals that with over 8 billion global mobile subscriptions, a substantial 84 EB/month of data traffic was exchanged. Looking ahead to 2027, projections suggest a significant increase, with 9 billion subscribers expected to exchange up to 368 EB/month of data traffic [1]. To meet ever-rising demands, mobile network operators need to expand network capacity. With LTE spectral efficiency nearing the Shannon limit, increasing capacity involves strategies like adding more cells, forming Heterogeneous and Small cell Networks (HetSNets) [2], or implementing techniques like multiuser Multiple Input Multiple Output (MIMO) [3] and Massive MIMO [4]. Though effective, these approaches result in higher inter-cell interference levels and increased costs. CAPital EXpenditure (CAPEX) rises due to the high cost of base stations in wireless network infrastructure while OPerating EXpenditure (OPEX) increases with significant power demand from cell sites (e.g., 72% of total power for China Mobile [5]). Mobile operators face the challenge of covering construction, operation, maintenance, and upgrade expenses, while Average Revenue Per User (ARPU) remains stagnant or decreases as users demand more data at lower costs. For instance, according to a report in [6] depicts scenarios where network costs surpass revenues (2014-2015). Hence, there is a critical need for innovative architectures to optimize both cost and energy consumption in the mobile network sector.

The potency of cloud computing in data storage, processing, service creation, and system operations has prompted mobile network operators to migrate their

functionality to the cloud. Initially, the focus was on core network services, as seen in initiatives, for instance, European Telecommunication Standards Institute (ETSI) Network Functions Virtualization (NFV) Industry Specification Group (ISG) [7]. Recent efforts are inclined towards addressing the baseband-processing needs of the radio access network (RAN) using high-volume IT hardware.

We discuss some of the key technological advances in the 5G mobile network that vastly expand the variety and manifoldness of network services to be supported.

Virtual RAN (VRAN)- Virtualization enables efficient resource sharing by creating isolated instances over abstracted physical hardware. In network evolution, V-RAN architecture [8, 9], distinct from C-RAN, prioritizes flexible control, cost-effectiveness, and diverse applications. V-RAN decouples software and hardware, supporting various services and minimizing operational costs, promoting energy efficiency and innovation for new market players at lower costs. While V-RAN has gained attention, scalability and latency limitations can be addressed by decentralizing it to the edge and leveraging fog computing, known as Mobile Edge Computing (MEC). In MEC, computing capabilities move closer to the radio access, ensuring low-latency, high-bandwidth access to content, applications, and services for subscribers. The distributed nature of MEC makes it ideal for handling large volumes of connected devices, a key focus in future wireless systems like 5G.

Open RAN (ORAN)- V-RAN is evolving towards O-RAN [10, 11] with a focus on Openness and Intelligence. Open interfaces support swift service introduction and create a competitive ecosystem. Open-source software and hardware designs enable faster innovation and deployment while ensuring backward compatibility. To meet the increasing complexity of future wireless systems like 5G, operators, and vendors must leverage self-organization and technologies like **Machine Learning and AI** for automated network functions and reduced operational costs. The open virtualized RAN is a crucial step toward 5G evolution, overseen by the O-RAN alliance, emphasizing open interfaces, RAN virtualization, and big data-enabled RAN intelligence with principles involving API specification, standards adoption, and the use of common-off-the-shelf hardware while minimizing proprietary hardware.

Mobile Edge Computing (MEC)- MEC [12] empowers the deployment of essential capabilities, including computing and network management, at the edge of

the mobile network, specifically within the RAN. By doing so, operators can execute core services close to end devices. This proximity not only enables application developers and content providers to offer context-aware services utilizing real-time radio access network information and user location but also enhances service responsiveness and diminishes bandwidth consumption. Crucially, MEC plays a pivotal role in achieving the ambitious technological benchmarks of 5G, such as a notable 1-millisecond latency, supporting over one million connections per square kilometer, and handling traffic rates ten times higher than those of 4G.

Network Slicing- Massive and highly heterogeneous network slicing [13] is a pivotal feature where tenants extend digitalization to consumers with services like holographic communication, multi-sensory experiences, and robotics. This entails the provisioning of numerous end-to-end slices traversing RAN, edge, cloud, and core domains addressing challenges such as low-latency communication, high data rates, and increased reliability. Network slicing allows the customization of logical networks atop a shared physical infrastructure, meeting diverse SLAs through isolation techniques [14]. In the realm of network slicing, 3GPP classifies 5G services into three categories: enhanced Mobile BroadBand (eMBB), ultra-reliable Low Latency (uRLLC), and massive Machine Type Communications (mMTC). This approach can optimize CAPEX/OPEX by efficiently sharing one physical infrastructure to meet the varied communication service requirements of emerging network services.

The thesis delves into leveraging these concepts to advance the state of the art in resource management and orchestration of edge services in 5G networks. More specifically, the investigation focuses on identifying techniques based on these concepts to ensure Service Level Agreements (SLAs) in the management of 5G networks.

The following sections will outline the research questions, key contributions, and the overall structure of the thesis.

1.1 Research questions and main contributions

The main contribution of this thesis is the design, development, and evaluation of resource orchestration architectures and management solutions for both MEC

applications and network services (RAN) for 5G/B5G networks with a focus both on theoretical-driven and ML-based decision-making. This work runs along two parallel tracks: **computationally aware network slice orchestration of radio resources** and **distributed orchestration of network and compute resources for edge services.** These both necessitate effective management of network and compute resources under dynamic conditions to meet various SLAs. A fixed network configuration proves insufficient for diverse network conditions and edge services unless resources are excessively overprovisioned. Optimal adaptation to varying data traffic, radio conditions, and resource availability is achieved through a dynamic network configuration, aligning services with specific needs without the need for resource overprovisioning. The distinction between these two parallel tracks lies in the methods employed for orchestration decisions—whether theoretically driven or based on machine learning. In the first case, a cost-efficient slicing strategy is formulated and solved, considering the practical correlation between the cost of computing resources at the network edge and the RAN's capability to support various network slices. While in the second case, network automation is executed at the edge of the network in a distributed manner, providing real-time and fine-grained local decisions.

In the rest of the section, a more in-depth description of the topics will be provided, specifying the research questions that have driven the research activity and our contribution to addressing them. We also provide details of our dissemination activity.

1.1.1 Characterizing the Computational and Memory Requirements of Virtual RANs

In the vRAN paradigm, a virtual evolved NodeB (eNB)/gNB centralizes a software-defined radio access stack within a computing edge infrastructure. The majority of the baseband processing of a virtual radio point of access (vRPA) is executed in virtual network functions (VNFs) operating on computing platforms, usually situated in proximity to the antennas. The enhanced flexibility offered by vRANs results in increased computing resource consumption at the network edge. This critical aspect has been inadequately addressed; most implementations neglect the demand for computational resources imposed by radio allocation, leading to inefficient pooling

of computing resources. It follows that the current efficiency of vRAN falls short of the optimal level, hindering its scalable deployment. To solve the above issues, it is critical to dynamically adapt the resource allocation to the temporal variations of the demand across vRPAs [15]. In this context, we have addressed the following main research questions (RQ).

RQ1:*What are the computing and memory requirements of a vRAN, as different settings in terms of number of occupied resource blocks (RBs) and type of modulation and coding scheme (MCS) are adopted?*

RQ2: *How do the computing and memory requirements of a vRAN change as the number of connected users (UEs) varies?*

We answer these questions by investigating the behavior of a vRPA using a test-bed implementation and conducting an extensive measurement campaign. In particular, we leverage a srsRAN implementation of an eNB and investigate its CPU and memory consumption under different experimental settings.

This part is covered in detail in Chapter 2. The dissemination of research work is published in peer-reviewed conferences and journals as listed :

1. S. Pramanik, A. Ksentini, C. F. Chiasserini, Characterizing the computational and memory requirements of virtual rans, in: 2022 17th Wireless On-Demand Network Systems and Services Conference (WONS), 2022, pp.1–8.

2. S. Pramanik, A. Ksentini, C. F. Chiasserini, Cost-efficient slicing in virtual radio access networks, Computer Communications 209 (2023) 349–358.

1.1.2 Cost-efficient Radio Access Network (RAN) slicing for service provisioning in 5G/B5G networks

Network slicing facilitates the creation of distinct logical networks, each tailored to specific network services, operating on a shared physical network infrastructure. This framework allows for the customization of slices to meet diverse SLAs. The trade-off between spectral efficiency and the consumption of data processing resources presents a complex scheduling dilemma. In essence, due to radio spectrum scarcity, efficient spectrum sharing is essential to fulfill the SLAs of each slice. Additionally, the limited computing resources at the network's edge highlight the significance of

allocating resources in a computationally aware manner across all slices. Further, the new features introduced by 5G new radio (NR) such as the concept of numerology, mini-slot based transmission, and punctured scheduling make the management of radio resources more complex. In this context, we address the above challenges, aiming at answering the following research questions (RQ):

RQ3: How can radio resources be sliced in a computationally aware manner to support traffic flows with different characteristics and QoS/SLA requirements?

To answer RQ3, we develop a model that captures the main aspects of a 5G RAN, and incorporates the relation that we were able to derive from our experiments between CPU utilization and number of users and radio resources allocated to the deployed slices. By exploiting such a model, we formulate and solve an optimization problem for resource usage reduction, while providing each slice with the requested QoS (namely, data rate and delay) in an isolated fashion. Specifically, our problem dynamically allocates resources to slices to maximize the profit of each slice, i.e., the difference between a slice utility (depending on its turn on the slice QoS requirements) and the CPU consumption due to the deployment of the slice itself on the vRAN. This part is covered in detail in Chapter 3

Our contributions can be found in:

1. S. Pramanik, A. Ksentini, C. F. Chiasserini, Cost-efficient slicing in virtual radio access networks, Computer Communications 209 (2023) 349–358.

1.1.3 Management and joint orchestration of virtual resources for MEC applications and network services (RAN).

In this work, we design an automated and efficient resource orchestration mechanism. With limited computational availability at the network edge, competition for resources between user applications and network services necessitates critical, automated resource orchestration in cases of scarcity. The computational demand of virtualized user applications and network service VNFs depends on data volume and their interdependence. This correlation, positive or negative, complicates the application of traditional modeling techniques for optimal resource allocation. Ex-

tending resource allocation approaches to multi-service scenarios poses significant scalability concerns, prompting the following research questions.

RQ4: How to provide fair and scalable orchestration of network and compute resources for edge services?

Specifically, how can resources be allocated to co-located edge user applications and network services within the same edge node, taking into account their potentially intertwined resource requirements?

We design a resource orchestration framework, named VERA to operate over an LTE vRAN and a user application, transmitting traffic through the vRAN to the mobile user, with a focus on providing scalable and fair resource allocation decisions. For certain user applications, the resource request between network functions and the application are closely interwind. Hence it mandates a joint allocation decision for optimal performance. VERA's performance is numerically validated on a proof-of-concept testbed. Additionally, we compare its scaling cost with that of a centralized framework based on deep-Q networks. A comprehensive analysis and evaluation of the VERA framework is presented in Chapter 4

Our contribution on this topic has been disseminated in:

1. S. Tripathi, C. Puligheddu, S. Pramanik, A. Garcia-Saavedra and C. F. Chiasserini, "VERA: Resource Orchestration for Virtualized Services at the Edge," ICC 2022 - IEEE International Conference on Communications.

2. S. Tripathi, C. Puligheddu, S. Pramanik, A. Garcia-Saavedra and C. F. Chiasserini, "Fair and Scalable Orchestration of Network and Compute Resources for Virtual Edge Services," in IEEE Transactions on Mobile Computing.

1.2 Outline

The thesis is organized as follows.

Chapter 2 focuses on the investigation and characterization of the computational and memory requirements of the RAN. First, it presents building a RAN test-bed utilizing the srsRAN open-source software-defined radio (SDR) LTE stack. Through

extensive experimentation, we thoroughly analyze the consumption patterns of computing and memory resources, evaluating overall system performance. Additionally, we establish regression models to anticipate system behavior with increasing connected users across various radio transmission settings. This methodology, along with the prediction models, contributes to the design and optimization of virtual RANs.

Chapter 3 presents CES, a cost-efficient slicing strategy designed for orchestrating radio resources to eMBB and uRLLC slices. CES is formulated considering the experimental findings of the CPU cost of RAN through a testbed implementation. Both orthogonal and non-orthogonal slicing methods are considered. CES performance is evaluated through numerical analysis, proving its capabilities in allocating radio resources in a cost-efficient way while also guaranteeing the target data rate and delay specified in the SLAs.

Chapter 4 presents VERA, a reinforcement learning framework designed for orchestrating resources for both user applications and network functions at the edge. The chapter begins by illustrating how the simultaneous resource requests from user applications and network functions are frequently interrelated. Subsequently, the advanced features of VERA are detailed, emphasizing the significant role of Pareto analysis in ensuring equitable decisions across diverse applications. Finally, the performance of VERA is rigorously evaluated through numerical analysis, substantiating its efficacy in meeting the designated Key Performance Indicators (KPIs).

Chapter 5 summarizes the presented work, highlighting the obtained results and concluding with key remarks and considerations.

Chapter 2

Characterizing the Computational and Memory Requirements of Virtual RANs

The virtualization of radio access networks (RANs) is a key component of future wireless systems, as it brings agility to the RAN architecture and offers degrees of design freedom. In this work, we investigate and characterize the computational and memory requirements of virtual RANs. To this end, we build a virtual RAN test-bed leveraging the srsRAN open-source mobile communication platform and general-purpose processor-based servers. Through extensive experiments, we profile the consumption of computing and memory resources, and we assess the system performance. Further, we build regression models to predict the system behavior as the number of connected users increases, under diverse radio transmission settings. In so doing, we develop a methodology and prediction models that can help designing and optimizing virtual RANs.

Part of the work described in this chapter has been already published in S. Pramanik, A. Ksentini, C. F. Chiasserini, "Characterizing the computational and memory requirements of virtual rans", in *17th Wireless On-Demand Network Systems and Services Conference (WONS),* 2022, pp.1–8.

S. Pramanik, A. Ksentini, C. F. Chiasserini, "Cost-efficient slicing in virtual radio access networks" in *Computer Communications* 209 (2023), 349–358

2.1 Introduction and motivation

Importantly, vRAN is a cornerstone technology for the realization of the emerging Open Radio Access Network (O-RAN) paradigm [10]. Indeed, the level of virtualization and flexibility that characterize a vRAN make it a perfect fit for the openness and intelligence concepts that are at the basis of the O-RAN architecture [16]. It is therefore expected that open standard radio frequency (RF) interfaces, combined with vRAN technologies, will further increase operational savings and increase the scalability of radio access networks.

However, the increased network flexibility and programmability allowed by vRANs come at the cost of a higher consumption of computing and memory resources at the network edge [17]. In particular, computing resources are typically pooled inefficiently since most of the implementations over dimension computing capacity to cope with peak demands in real-time workloads [18]. It follows that the gains currently attainable by a vRAN are far from optimal, preventing its deployment at scale.

To solve the above issues, it is critical to dynamically adapt the resources allocation to the temporal variations of the demand across vRPAs [15]. Elastic slice-aware computational resource management algorithms [19], play a pivotal role in optimizing resource utilization and maximizing cost efficiency. A key initial measure for efficient computational resource management involves estimating the processing resources depending on various radio settings such as Modulation and coding schemes (MCS), number of occupied resource blocks (RBs) and number of connected users. This estimation not only enhances the slice optimization process but also contributes to achieving elevated computational resource utilization. To comprehend, these findings will provide insights for determining the ideal configurations of vRAN parameters when faced with limitations in CPU and memory resources at the network edge.

Towards this goal, a first, fundamental step is to gain a better hands-on understanding of the behavior of vRPAs and the relation between radio and computing/memory resource dynamics, as well as their dependency on such factors as radio channel conditions and user's traffic demand. While the studies in [15, 20–23, 18, 24–26, 17, 27] have focused on the optimization of vRANs through experimental work or by designing analytical models and algorithmic solutions, to our knowledge, this paper is

the first attempt to characterize the computing and memory resource consumption of vRANs under diverse settings, as the number of connected users increases.

We answer these questions by investigating the behavior of a vRPA using a test-bed implementation and conducting an extensive measurement campaign. In particular, we leverage an srsRAN implementation of an eNB and investigate its CPU and memory consumption under different experimental settings. It is worth noting that CPU utilization is a key metric used to track the system performance behavior, however modern processor technology is much more complex, as a single processor package may encompass multiple cores with dynamically changing frequencies. These technological advances can thus change the behavior of CPU utilization reporting mechanisms. Nevertheless, our analysis is carried out in the same environment, which will not only provide qualitative insights but quantitative predictions as well.

Our main contributions are as follows.

- We develop an srsRAN-based experimental test-bed and perform extensive experiments, in order to profile the performance limits of the eNB in terms of processing, memory, and throughput. We show that the CPU utilization of the eNB increases with the modulation and coding (MCS) index, number of occupied Resource Blocks (RBs), and importantly, with the number of connected users.

- Using empirical data, we define regression models to predict the percentage of CPU utilization and memory consumption of the virtual eNB, as the number of connected users varies. In so doing, we obtain a prediction accuracy of 99% for both CPU and memory utilization. These approximated models provide real-world insights and key inputs to formulate, design, and evaluate optimized resource management in vRANs.

The rest of the paper is organized as follows. Section 2.2 introduces the design and implementation of our vRAN test-bed, while Section 2.3 presents experimental results and our empirical models. Section 2.4 discusses the related work and highlights the novelty of our study. Finally, Section 3.7 concludes the paper and presents possible directions for future research.

(a) (b)

Fig. 2.1 vRAN test-bed implementation using srsRAN: (a) snapshot of our test-bed highlighting the edge node hosting the virtual eNB and EPC, and four UEs; (b) test-bed architecture including UEs, virtual eNB and virtual EPC

2.2 vRAN Test-bed

We now introduce our vRAN test-bed using srsRAN, detailing the test-bed architecture and configuration, and the adopted experimental methods.

2.2.1 Test-bed architecture

Figure 2.1(a) provides a snapshot of the test-bed we developed, while Figure 2.1(b) represents its architecture. We leverage software-defined radio (SDR) interfaces enabling point-to-point communications between vRPA and User Equipment (UE). A vRPA implements the necessary processing stack to transfer data to/from UEs. In our case, the vRPA acts as a virtual eNB implemented at the edge of the network. The connectivity between the vRPA and UEs is supported by means of an LTE radio link implemented using the srsRAN [28] – an open-source SDR LTE stack implementation offering Evolved Packet Core (EPC), eNB, and UE applications. It is compliant with LTE Release 9 and supports up to 20-MHz bandwidth channels as well as transmission modes from 1 to 4, all using the frequency division duplexing (FDD) configuration.

As RF front-end, Ettus Universal Software Radio Peripheral (USRP) B210 devices are used to perform up/down-conversion, filtering, amplification and AD/DA conversion of the UE and eNB LTE signals. All the RF front-ends are connected to the vRPA and the UEs via USB 3.0. Then, physical layer is implemented through a set of OFDMA-modulated channels, using RB filling across ten 1-ms subframes forming a frame. RBs assigned to UEs by the MAC layer are modulated and encoded

with an MCS that depends on the user's Channel Quality Indicator (CQI), a measure of SNR that is locally available at the UEs for uplink transmission and is reported periodically by UEs. The modulation order can vary up to 256-QAM, while FEC is employed using LDPC or Turbo codes with coding rate of 1/2, 2/3, or 3/4 [29]. At the MAC layer, an automatic repeat request error control is in place, i.e., an unsuccessfully transmitted packet can be resent till a maximum number of allowed re-transmission attempts.

The edge host and the mobile terminals are each installed in Ubuntu 18.04 systems. The edge host is equipped with an Intel i7-7700HQ 4-cores CPU and 8 GB of DDR4 RAM, while the UEs feature an Intel i7-8550U 4-cores CPU and 16 GB of DDR4 RAM. Each Ubuntu system is connected to USRP B210 boards using USRP Hardware Driver v3.15. In order to faciltate the experiments, all performance management features in the BIOS (e.g., Intel@TurboBoost, Hyper-thread control, Intel SpeedStep) are enabled and C-states have been turned off. The CPU governor of the edge host and the UEs are set to performance mode to allow for maximum computing power and throughput. Moreover, the real-time thread priorities are enabled in the srsRAN as the applications (srsENB and srsUE) are executed with root privileges. A set of threads are created in srsRAN for performance and priority management reasons. The threading architecture of the physical layer implementation is motivated by the stringent latency requirements. Also, we monitor the level of CPU consumption and ensure that, during our experiments, the allocated CPU is sufficient to keep up with the required data rate so as to avoid severe system failures during the radio data transfer. Finally, in order to establish a stable connection, we set the transmit gain (tx_gain) at the eNB to its maximum value.

2.2.2 Monitoring the srsRAN eNB and UEs

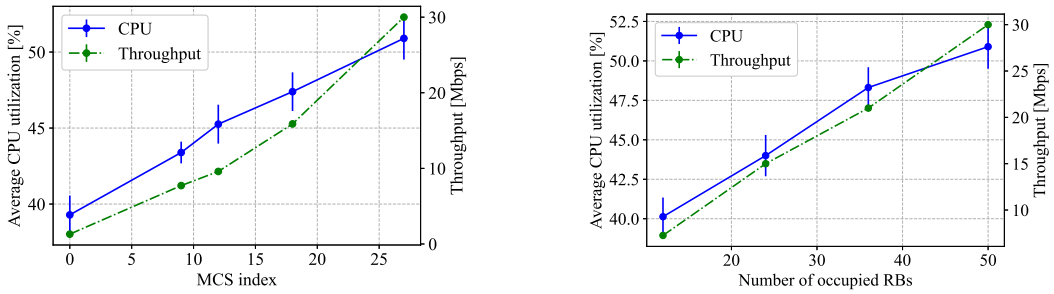
To monitor the behavior and track the performance of the vRAN entities, we use some of the useful features of srsRAN (e.g., detailed log system with per-layer log levels, MAC layer Wireshark packet capture, command-line trace metrics, detailed input configuration file). A configuration file is provided to set parameters such as downlink carrier frequency and log or packet capture options, making the software easy to use. Moreover, we have relied on a useful feature of srsUE which provides real-time traces.

The srsENB application metrics are generated once per second by default. Metrics are provided on a per-UE basis for the downlink and uplink, respectively. The eNB is configured in band 7 (FDD) and the transmission bandwidth has been set to 10 MHz, corresponding to 50 RBs. In order to determine the successful connection between eNB and UE, the RRC states are observed. Specifically, when the UEs are successfully paired to the eNB, the RRC connection setup message is seen. We have also saved the UE and eNB logs to verify such entities' status.

As experimental set-up, we connected 30 dB attenuators to the antennas of each network node; furthermore, the UEs were placed close enough to the eNB so as to ensure high values of SINR ($\geq$ 25 dB). We focus on downlink data transfer and used iperf for data packet generation.

2.3 Experimental Evaluation and Analysis

In this section, first we present the performance of the vRPA, i.e., the srsRAN eNB, in terms of CPU utilization as the number of occupied RBs and the MCS index vary, when a single UE or multiple UEs are connected. A similar evaluation for the memory utilization can be found in our conference paper [30]. The results have been obtained by averaging over 10 experiments; in every plot, both the average value of the presented performance metric and the corresponding 95% confidence interval are shown.



(a) (b)

Fig. 2.2 CPU utilization and UDP downlink throughput of the virtual eNB, for different values of the MCS index (a) and occupied RBs (b), and a single connected UE

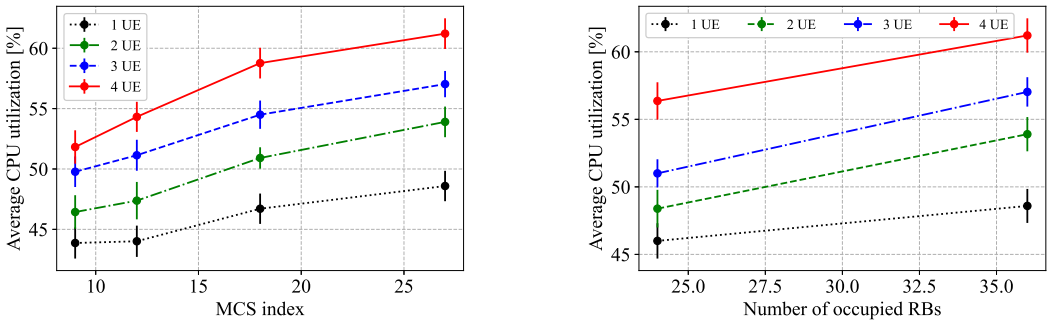
2.3.1 CPU utilization

The CPU usage is analyzed using an Intel Core i7-7700HQ 2.80 GHz CPU. It is calculated using the `top` process in Linux, which provides a dynamic real-time view of a running system managed by the kernel. Specifically, the percentage of consumed CPU is collected by sampling the `top` reports every second, over a period of 200 seconds.

Figure 2.2(a) shows the average CPU utilization (left-hand side y-axis) and the throughput (right-hand side y-axis) obtained over 10 iterations of the virtual eNB, as the MCS varies and for a single connected UE. For this experiment, UDP DL traffic is generated at the eNB at 30 Mbps, setting the number of allocated RBs to 50. Also, we set the transmission gain to its maximum value, thus ensuring that the SNR does not drop below 32 dB. From the plot, we can observe that, as also shown in [17], the CPU utilization of the virtual eNB increases as the MCS index grows from 0 to 27. Further, the consumption of computing resources, which is mainly due to the modulation, demodulation, coding and decoding operations, is quite significant in absolute terms: as an example, for MCS$=27$, a single user consumes around 51% of a single CPU of the edge node. Using empirical data, we found that the CPU utilization of the virtual eNB can be well approximated as a linear increasing function of the MCS, i.e., $\text{CPU}[\%] = 0.429 \cdot \text{MCS} + 39.58$.

Figure 2.2(b) shows the CPU utilization (left-hand side y-axis) of the virtual eNB as the number of allocated RBs varies from 12 to 50, for a single UE and MCS$=27$. The DL traffic load is set to 9 (for 12 RBs), 15 (for 24 RBs), 21 (for 36 RBs), and 30 (for 50 RBs) Mbps, respectively. We notice that the CPU utilization increases as the number of occupied RBs increases, with a maximum of 51% for a single UE. A higher number of occupied RBs leads to the user transmitting at a higher rate, which results in a higher computational resource consumption. From the experimental data, we found that the CPU utilization of the eNB can be well approximated as a linear increasing function of the number of allocated RBs, a, i.e., $\text{CPU}[\%] = 0.2892 \cdot a + 37.02$. We remark that the provided approximation functions can help interpolate the average CPU utilization with different radio configurations.

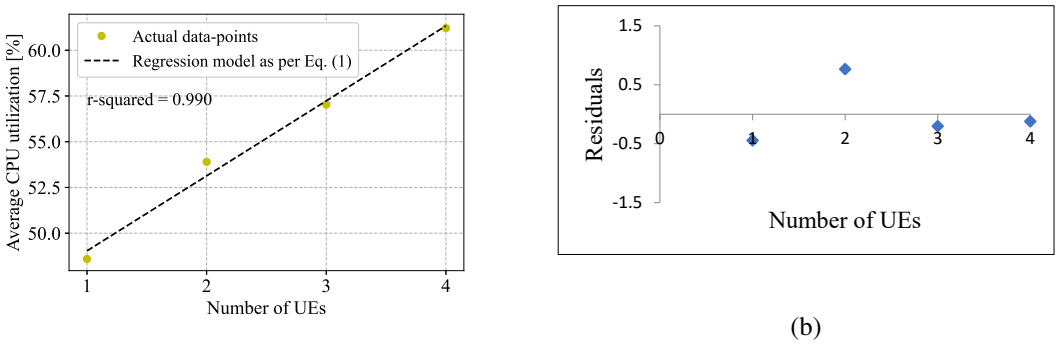
We are now interested in how the computing resource consumption varies as the number of users connected to the eNB changes. It is indeed a fact that the number of served UEs is rapidly increasing, and that cellular networks will have to support a



(a) (b)

Fig. 2.3 CPU utilization of the virtual eNB for a varying number of connected UEs, versus the MCS index and number of occupied RBs equal to 36 (a), and versus the number of occupied RBs and for MCS= 27 (b)

massive number of users. Figure 2.3(a) presents the CPU utilization of the virtual eNB as the MCS index varies, for different numbers of users. For this experiment, the overall maximum number of RBs that can be used is set to 36, downlink traffic is generated at 21 Mbps, and the tx_gain is set to its maximum value, so that the SNR is always above 28 dB for all the UEs. In this scenario, an interesting behavior emerges: for a fixed value of the MCS index, the average CPU consumption of the eNB increases significantly as the number of users increases, although the traffic load is kept constant. As an example, for MCS= 27, the average CPU consumption with four UEs is 62% of a single CPU, i.e., about 30% more than with one UE.



(a) (b)

Fig. 2.4 (a) Regression plot as the number of UEs varies, for 36 occupied RBs and MCS= 27, (b) Residual plot for the regression model in (2.1), with 36 occupied RBs and MCS= 27

In addition, Figure 2.3(b) shows the CPU consumption of the virtual eNB as the number of allocated RBs varies from 24 to 36, for a different number of connected UEs, MCS= 27, and maximum tx_gain. The DL traffic load for 24 RBs is set to

15 Mbps, while it is 21 Mbps for 36 RBs, so that all the allocated RBs are always occupied. Interestingly, as the number of connected users grows, the CPU consumption of the virtual eNB increases linearly.

Since the MCS index and number of occupied RBs are always finite values that vary over a very specific range, we are mainly interested in understanding the CPU requirements of the virtual eNB as the number of UEs increases. Then, using the empirical data, we build a regression model that predicts the CPU utilization of the eNB as the number, n, of connected UEs varies. For 36 occupied RBs and MCS $= 27$, we obtain:

$$CPU[\%] = 4.099 \cdot n + 44.935\,. \tag{2.1}$$

We remark that similar models can be built for different values of MCS index and number of occupied RBs.

Figure 2.4(a) shows the CPU utilization under the above settings, along with the curve obtained using the regression model in (2.1). The Significance F for our model is 0.004, which, being well below 0.05, shows that the model can predict correctly the behavior under study. Further, the regression output (R-squared) indicates that 99% of the variation in CPU consumption is due to the number of UEs. Specifically, every additional UE is expected to entail about 4.1% of increase in CPU usage at the eNB; it follows that 15 users will easily consume up to 100% of a CPU. [1]

Again for 36 occupied RBs and MCS $= 27$, Figure 2.4(b) plots the residuals, i.e., the difference in percentage between the actual value of CPU utilization and the one predicted by the regression model, obtained when the number of UEs varies between 0 and 4. We observe that such residuals are always within -0.5% to 1% of CPU usage, which confirms the very good accuracy of the model.

Next, it is important to show that a linear regression model (as in (2.1)), obtained from experimental data, correctly characterizes the computing requirements of a vRAN as the number of users increases. To this end, we use all the experimental data obtained for a number of UEs up to 3 (i.e., for a varying number of occupied RBs and MCS indices), and we predict the CPU consumption when four UEs are

[1] %CPU in top reports the task's share of CPU time as a percentage of total CPU time. In an SMP (systems with multiple processors) environment with multi-threaded processes %CPU values exceeding 100% may be reported due to parallel processing across multiple CPU cores.

connected. The corresponding regression model is given by:

$$CPU[\%] = 3.9 \cdot n + 0.369 \cdot \text{MCS} + 35.658 \tag{2.2}$$

where n is the number of connected UEs and MCS is the adopted MCS index. Similarly, the CPU consumption as a function of the number of connected UEs (n) and number of allocated RBs (a) can be written as:

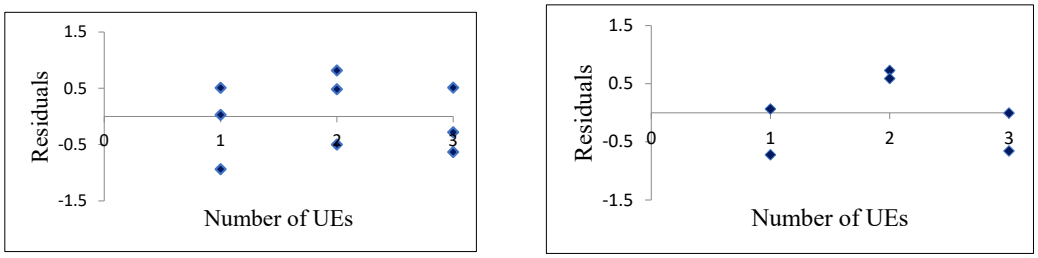
$$CPU[\%] = 3.9 \cdot n + 0.44 \cdot a + 30\,. \tag{2.3}$$

Table 2.1 Actual vs. predicted CPU usage with 4 UEs, a varying number of occupied RBs, and MCS = 27

No. allocated RBs	**Actual CPU [%]**	**Predicted CPU [%]**
24	56.36	55.51
36	61	60.759

Table 2.2 Actual vs. predicted CPU usage with 4 UEs, different MCS indices, and 36 occupied RBs

MCS index	**Actual CPU [%]**	**Predicted CPU [%]**
12	54.31	55.55
18	58.76	57.71
27	61	61.19



(a) (b)

Fig. 2.5 (a) Residuals plot for the regression model in (2.2), for different values of the MCS index and 36 allocated RBs; (b) Residuals plot for the regression model in (2.3), for a different number of allocated RBs and MCS = 27

Table 2.1 and Table 2.2 report the actual CPU utilization when 4 UEs are connected, and the corresponding value predicted through (2.2) and (2.3). Furthermore,

Figure 2.5(a) plots the residuals obtained for the model in (2.2) and Figure 2.5(b) those obtained for the model in (2.3): again, all residual values are within -1% and 1%. These results, along with an F-statistic value of 0.0023, indicates that the linear regression model well describes the behavior of CPU utilization.

The experiments are performed with at least one connected UE. For MCS= 27 and RBs= 36, the predicted CPU utilization from (2.2) is 45.62%, which is very similar to the value of 45.84% obtained from (2.3) with no connected UE. Further, in Fig. 2.2(a) and Fig. 2.2(b) we validate our models in (2.2) and in (2.3) with the linear functions provided for one UE, as functions of the MCS (varied from 0 to 27) and the number of allocated RBs (varied from 12 to 50), and considering as minimum values of the MCS and the number of allocated RBs 0 and 12, respectively. We can observe that the CPU usage is almost identical with the same parameter settings for both models, which confirms the accuracy of the prediction models in (2.2) and (2.3).

In order to utilize the linear model with diverse parameter settings (different MCS and RB configurations), all our experimental data can be exploited to derive a model as the number of UEs (n), MCS, and RBs vary. The corresponding regression model is given by,

$$CPU[\%] = 3.46 \cdot n + 0.325 \cdot \text{RBs} + 0.28 \cdot \text{MCS} + 26.55 \tag{2.4}$$

For MCS= 27 and number of occupied $RBs = 36$, the predicted CPU usage from the model in (2.4) is 59.65% for 4 connected UEs, while the actual CPU usage from the experiments is 61% for the same parameter settings. Figure 2.6 plots the residuals obtained for the model in (2.4) showing that all residual values are within -1.5% and $+1.5\%$. These results, along with an F-statistic value lower than 0.05, indicate that the linear regression model well describes the behavior of CPU utilization. It is worth mentioning that the CPU utilization at the eNB can be accurately predicted also when different values of MCS are used for different UEs. Indeed, since the MCS index takes discrete values over a very specific range, using in (2.4) the average value of the MCS indices allocated over the different users still provides an accurate estimate of the CPU consumption at the eNB.

Finally, it is important to underline that the relationship between CPU consumption and factors such as the MCS, the number of allocated RBs, and the number of connected UEs may become non-linear for a certain number of UEs. Although, due

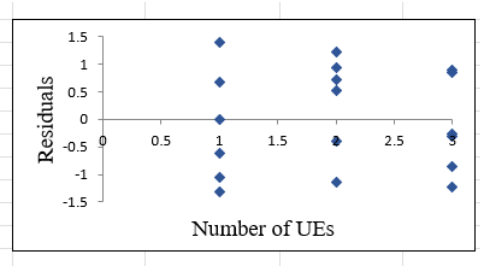
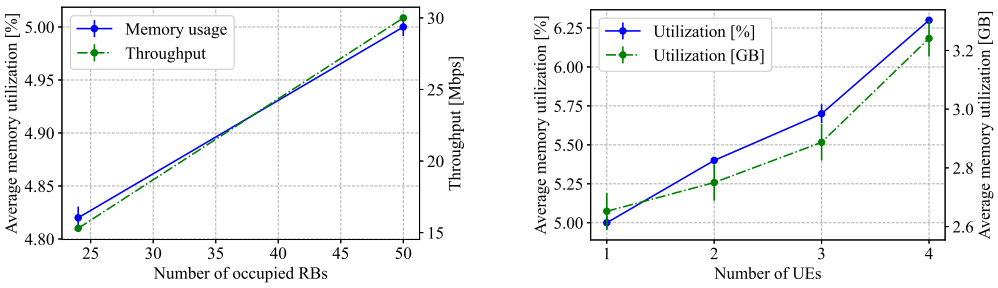


Fig. 2.6 Residual plot for the regression model in (2.4), for different values of the MCS index and occupied RBs

to hardware constraints, we had to restrict our analysis to 4 UEs, we have profiled the downlink scheduler when UEs are varied from 1 to 4 and it was found that the processing time increases with the number of users: as an example, the downlink scheduler processing time for 1 UE is 10 μs, while for 4 UEs it is 20 μs. Additionally, some recent work [27] confirms that the processing time of downlink transmission tasks does vary with the number of users (e.g., scheduling becomes more complex). Based on our experiments, we observed that under low values of MCS (as shown in Fig. 3(a)), the primary CPU consumption is not due to the MCS, but rather to the increment in the number of users. For instance, for MCS$=9$, the CPU consumption is 43.63% for 1 UE, while it is 52% for 4 UEs. For MCS$=27$, the CPU consumption is 48.58% for 1 UE, while it is 61.209% for 4UEs.

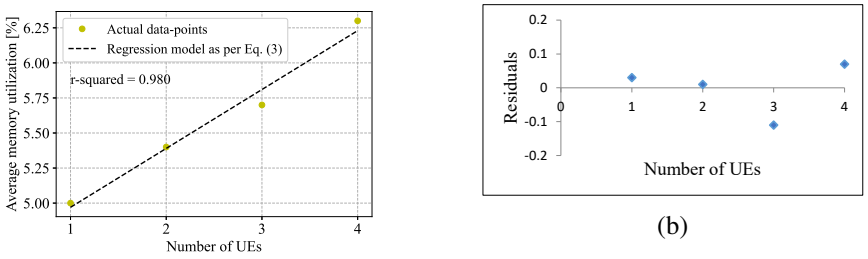
To conclude, our proposed linear relationship models hold for the downlink[2] traffic transfer with any value of MCS between 0 and 27, occupied RBs from 0 to 50 (i.e., for a 10 MHz channel), and a number of UEs from 1 to 4. From our experiments, we found that, for higher values of MCS index (i.e., more sophisticated modulation and coding schemes are used), number of allocated RBs (i.e., higher-rate transmissions take place), and number of users, the CPU consumption increases linearly. Moreover, the dominant impact on CPU consumption is due to the number of connected UEs. Complexity of assessing CPU performance is not only based on hardware parameters like clock speed and the number of cores, factors influence CPU performance beyond these hardware specs. CPUs with superior performance (attributed to higher clock speeds or more usable cores) may exhibit lower utilization.

[2]It is worth observing that uplink data transfer will unarguably influence the analysis of the CPU utilization: e.g., the processing time for decoding is longer than for encoding, and it increases with the MCS index, as shown in [17, 15, 27].



(a) (b)

Fig. 2.7 Memory utilization of the virtual eNB (a) as the number of occupied RBs varies, for MCS= 27 and 1 connected UE, (b) vs number of UEs, for MCS= 27 and 50 occupied RBs



(a) (b)

Fig. 2.8 Percentage of memory usage: (a) linear regression plot, (b) residual plot for MCS= 27 and 50 occupied RBs.

However, in a stable environment, the relationship between CPU utilization and radio parameters remains consistent, regardless of the specific hardware configuration.

2.3.2 Memory Utilization

Besides CPU, also memory is a precious resource at the network edge. It is therefore important to investigate the memory usage of the virtual eNB for different radio configurations.

The memory usage is monitored using the `top` process in Linux, which reports both the amount of memory used by the process in GB as well as the percentage value. As done for the CPU utilization, we profile the memory usage for different values of MCS index, number of occupied RBs, and number of connected UEs. In our experiments, the downlink traffic load is set to 30 Mbps in order to ensure that all allocated RBs are actually occupied.

Figure 2.7a shows the memory usage (left-hand side y-axis) as the number of occupied RBs varies from 24 to 50, for MCS= 27 and only one connected UE. As observed for the CPU utilization, the memory usage also increases linearly with the number of occupied RBs. We also find a similar behavior as the number of UEs grows, as shown in Figure 2.7b for MCS= 27 and 50 occupied RBs. In particular, the percentage of memory utilization (left-hand side y-axis) increases almost linearly with the number of users, in spite of the constant traffic load. Furthermore, we observe that 3.24 GB of memory are consumed with four connected UEs, which is over 20% more than when only one UE is connected.

Based on the experimental data we obtained, the regression model predicting the percentage of memory utilization, for 50 occupied RBs and MCS= 27, is as follows:

$$Memory[\%] = 0.42 \times n + 4.55, \tag{2.5}$$

where n is the number of UEs. A similar linear regression model can be built for different values of number of occupied RBs and MCS index.

The accuracy of the model is shown in Figures 2.8a–2.8b, for MCS= 27 and 50 occupied RBs. The two plots depict, respectively, the difference between the actual and predicted percentage of memory utilization, and the residuals, as the number of UEs varies. Both figures highlight that also the regression model we built for the percentage of memory utilization closely mimics the resource consumption of a vRAN.

2.4 Related Work

Owing to the intricate relationship between radio and computing resource dynamics, and the advantages offered by vRANs, several works have aimed at investigating and optimizing such a virtual system. While the studies in [15, 20–23, 31] focus on evaluating the performance of a vRAN through experiments, the works in [18, 24–26, 17, 27] provide an insight onto the theoretical framework.

More specifically, [15] is one of the first experimental studies that characterize the potential savings in compute resources when exploiting the variations in the processing load across base stations. Interestingly, [20] presents a linear model to calculate the uplink processing time for a single user in terms of the sub-carrier load,

MCS index, and number of antennas. The linear model is then used to develop RT-OPEX, a C-RAN scheduling algorithm. The impact of MCS and SNR on real-time C-RAN processing (i.e., CPU) is studied in [21], along with a mathematical model for predicting the decoding time. The work in [22] profiles instead the performance of a C-RAN in terms of CPU and memory usage, as the iperf transmission bandwidth increases. In [23], the authors investigated the CPU consumption of the baseband unit (BBU) under various conditions for the C-RANs, and characterized the computational demand in terms of throughput. Instead, the work in [31] introduces a processing time model considering the MCS, the number of RBs, and the CPU frequency. However, no such work has investigated and characterized the performance of a vRAN in terms of CPU and memory usage as the number of connected users increases and under diverse settings.

As far as analytical models and algorithmic solutions for the optimization of a vRAN are concerned, [18, 32] set a theoretical basis for CPU-aware radio resource control. In particular, [18] aims at reducing the level of variability of the computational load, by jointly optimizing the selection of the MCS index and the allocation of the physical resource blocks (PRBs). [24, 33] investigate instead the trade-off between the consumption of data processing resources and achievable data rates, taking into account specifically the processing requirements of forward error correction (FEC) on the uplink. A computationally aware MCS selection policy is proposed that reduces the computational complexity requirements, at the cost of slightly decreased spectral efficiency in [24]. The above works rely on the same model relating computational requirements and SNR, and they neglect variations on the arrival bit-rate load. This issue is addressed instead in [34], which combines real-time traffic classification and CPU scheduling in a mobile edge computing setup. However, [34] relies on a simplistic base-band processing model and does not include an experimental validation. An analytical framework, FuidRAN, is presented in [25], which jointly selects the function split and routing policy, tailored to the available network and computing resources. However, the model is provided for one user only. In [26], a novel reinforcement learning framework is presented, which efficiently allocates radio resources to multiple users in terms of link, MCS index, RBs and airtime for packet transmissions in heterogeneous vRANs. A related relevant contribution is also given in [17] where the vrAIn solution that dynamically learns the optimal allocation of computing and radio resources in order to meet the target level of QoS. A novel pipeline architecture for 5G distributed units (DUs) is

presented in [27] to guarantee a minimum set of signals that preserve synchronization between the DU and its users, during computing capacity shortages. This study relies on techniques that require predictable computing to provide carrier-grade reliability.

In conclusion, no previous work characterizes the computing requirements of vRANs with respect to complete contextual dynamics (namely, traffic load and number of users). Importantly, designing resource allocation schemes ignoring such requirements may severely hamper the system performance in terms of throughput.

2.5 Conclusions and Discussion

Virtualized radio access networks (vRANs) are the basis of future base stations design. To provide real-world insights and key inputs to formulate, design, and evaluate optimized resource-management problems in vRANs, we investigated and characterized the computational requirements of vRANs by developing an srsRAN-based test-bed. Through extensive experiments, we profiled the CPU and memory utilization of the vRAN. Our results shed light on the vRAN behavior across different scenarios, showing that, remarkably, the CPU and the memory utilization of the eNB increases substantially with the number of users. It is worth underlining that the result has been obtained under a constant value of traffic load and number of occupied resource blocks. Based on these empirical results, we also built linear regression models for the prediction of CPU and memory utilization as the number of users varies. To the best of our knowledge, this is the first work that thoroughly studies the computational and memory requirements of a vRAN.

System (hardware) parameters, e.g., clock speed and number of cores, are not always directly related to performance, as more factors may affect a CPU performance. CPUs that perform better (either by higher clocks or with more usable cores) will have lower utilization. The threading architecture in srsRAN is efficient for dual- or quad-core CPUs. If more than 4 cores are available, increasing the number of physical threads above 3 does not add any benefit, because the maximum tolerable latency is 3 ms (3 pipeline stages). Moreover, the rate of variation in CPU utilization with respect to the radio parameters will remain the same in an unchanged environment. Clearly, the higher the number of physical cores, the more users can be supported. Though the analysis considered the impact of traffic volume, the evaluation could be extended to considering fluctuating radio conditions. We believe

that our study and the obtained models can provide researchers and practitioners with real-world insights and the necessary tools for designing advanced efficient resource-provisioning and allocation strategies in vRAN systems. As future work, we intend to exploit the results of this work to develop algorithms for the efficient and effective scaling of the vRAN functions, and the optimal settings of the vRAN parameters in the case of shortage of CPU and memory resources at the network edge.

Chapter 3

Cost-efficient RAN Slicing for Service Provisioning in 5G/B5G

Network slicing represents a substantial technological advance in 5G mobile network, greatly expanding the variety and manifoldness of network services to be supported. Additionally, 3GPP 5G New Radio (NR) has introduced novel features such as mixed numerology and mini-slots, which can be harnessed by network slicing to cater to the diverse requirements of 5G services. While however the co-existence of multiple network slices leads to a challenging resource allocation problem, these new features also severely complicate the management of radio resources. As a further point of attention, the virtualization of radio functions may exact a significant toll from the, already limited, computing resources at the network edge. It follows that a cost-efficient resource allocation across all the slices becomes crucial.

The first part of this work considers an orthogonal slicing approach. In the second part, a non-orthogonal slicing approach is contemplated to leverage the innovative features introduced by 5G NR, as elucidated earlier. By using the numerology and mini-slot based transmission different transmission time intervals (TTIs) can be supported, allowing for packet transmission over a short period for uRLLC users with stringent delay requirements. Leveraging on our experimental findings (discussed in Chapter 2), a cost-efficient network slice dimensioning problem is formulated, named cost-efficient slicing (CES) which maximizes the difference between total utility and CPU cost of network slices. To reduce the complexity of the second approach, it is decomposed into two sub-problems, namely, the scheduling problem

of eMBB UEs on a time-slot basis and of uRLLC UEs on a mini-slot basis while keeping the objective unchanged. To address the scheduling issue of eMBB UEs, a heuristic technique is employed, and, using the outcome of this heuristic, the optimal solution for the problem of uRLLC UEs is derived. The significance of the proposed approach over a baseline approach is evaluated through numerical simulations. The comparison factor is the number of allocated eMBB and uRLLC RBs per time-slot for the orthogonal slicing approach while for the non-orthogonal slicing is the number of occupied uRLLC RBs per mini-slot. An assessment is performed by measuring the impact of the uRLLC slice changes on the eMBB slice, and vice versa, including delay for uRLLC users and data rates for eMBB users.

The first part of the work described in this chapter has been published in S. Pramanik, A. Ksentini, C. F. Chiasserini, "Cost-efficient slicing in virtual radio access networks", *Computer Communications* 209 (2023), pp. 349–358.

The second part of the work described in this chapter is under review in *Computer Communications"*

3.1 Introduction

Massive and highly heterogeneous network slicing is a key feature of beyond-5G and 6G networks (B5G/6G), where tenants are not only targeting vertical industries but also extending digitalization to the final consumer through new services such as holographic communication, multi-sensory experience, and robotics. In this respect, 6G networks need to handle massive end-to-end slices that cross multiple technological domains — i.e., radio access network (RAN), edge, cloud, and core, and effectively address the challenges they pose in terms of low-latency communication, high data rate, and increased reliability.

Network slicing enables multiple logical networks corresponding to different network services running on top of a common physical network infrastructure, with the possibility to customize slices to satisfy various service level agreements (SLAs) through isolation techniques [14]. Considering the concept of network slicing, 3GPP has classified 5G services into three distinct classes according to their communication service requirement: (i) enhanced Mobile BroadBand (eMBB), (ii) ultra-reliable Low Latency (uRLLC), and (iii) massive Machine Type Communications (mMTC).

In this context, network slicing can help to reduce CAPEX/OPEX as one physical infrastructure is shared efficiently to fulfill the heterogeneous communication service requirements of emerging network services.

Although network slicing is well-researched, slicing (sharing) the RAN resources is still challenging. The new features introduced by 5G new radio (NR) such as the concept of numerology [35], mini-slot based transmission [36], and punctured scheduling [37] make the management of radio resources further complex. Numerology entails the provision of various frequency domain subcarrier spacings (SCSs) and time domain symbol lengths within the time-frequency OFDM grid. The flexibility in numerology allows for efficient scheduling of eMBB and uRLLC users by choosing SCS and OFDM symbol lengths that align with service requirements. Additionally, the mini-slot approach supports transmission shorter than the regular slot duration. A mini-slot (or the smallest scheduling time unit) occupies 2, 4, or 7 OFDM symbols (regardless of numerology). Finally, the punctured scheduling enables non-orthogonal slicing of radio resources and facilitates the uRLLC traffic to preempt resources that have already been allocated to the eMBB users. Taking into account these three techniques, and their potentiality in fulfilling service requirements, makes the RAN slicing a multi-timescale, non-trivial problem.

As an example, using a numerology (μ), the time duration of the PRB is scaled down by a factor 2^{μ} while the frequency is scaled up by 2^{μ}. Thus, using higher numerology and shorter mini-slot duration decreases the RAN latency, but it increases the amount of processing, hence the system energy consumption, since UEs and gNB execute several RAN functions 2^{μ} more times per time unit. The trade-off between spectral efficiency and the consumption of data processing resources presents a complex scheduling dilemma. To elaborate, the scarcity of radio spectrum necessitates efficient spectrum sharing to meet the SLAs of each slice. Simultaneously, the limited computing resources at the network's edge underscore the importance of allocating resources in a computationally aware manner across all slices. If the service in a slice has elasticity [38], then the resource demand of the slice can change depending upon the operational computational cost, to maximize the slice profit. This inspires us to deeply explore the relationship between computing resource cost and slice dimensioning. While the state-of-the-art on 5G RAN slicing [39–44] mainly focuses on offering a satisfying level of quality of service (QoS) or user's quality of experience(QoE), none of the existing works designs a cost-efficient slicing strategy

accounting for the real-world dependency between the cost of computing resources at the network edge and the ability of the RAN to support different network slices.

The key contributions to this work are:

- The problem of cost-efficient resource management is addressed by developing a model that captures the main aspects of a vRAN and incorporates the relation that we were able to derive from our experiments between CPU utilization and number of users and of radio resources allocated to the deployed slices (discussed in Chapter 2 of our work).
- By leveraging such a model, an optimization problem is formulated that aims at maximizing the slice profit. Such profit is defined as the difference between the sum of the utility of all eMBB user equipments (UE) across t time-slots and the normalized cost of computing resource consumption due to the slices supported on the RAN.
- Non-orthogonal slicing is also formulated as a *mixed-integer quadratically constrained program* (MIQCP) problem. However, in addition to throughput and latency requirement, further scenarios, for instance, different values of numerology, mini-slot durations per slice are considered.
- In light of the problem complexity of the latter, the original problem (P) is decoupled into two sub-problems, one tackles the resource allocation for eMBB UEs on a time-slot basis (P1), and the other addressing the resource allocation for uRLLC UEs on a mini-slot basis (P2). The first sub-problem is redefined into a maximization problem for each time slot, and the second sub-problem is a maximization problem for each mini-slot within every time-slot.
- Due to the NP-hardness of P1, a low-complexity heuristic is envisioned to solve it, thus improving the minimum expected achieved rate (MEAR) among eMBB users (providing the eMBB users with the target). Next, an M/M/1/k queue is used to model the delay of the uRLLC users, and a utility function for eMBB users to represent the network resources utilization and the target data rate. In so doing, P2 is reformulated taking into account both the decision made by solving P1 and the computing cost associated with the slices. Finally, at every time slot, the new formulation of P2 is solved to maximize the efficiency in resource utilization, while meeting the target eMBB data rate and uRLLC delay.

- A comprehensive experimental analysis is performed for the proposed scheduling approach. A comparative analysis of the results of non-orthogonal slicing is presented in terms of an average number of occupied uRLLC resource blocks (RB)s per mini-slot and average delay of uRLLC UEs, against the Static Resource Slicing (SRS) approach [45, 42]. The slice requests in SRS are processed without considering the CPU cost of the gNB due to slicing. On the contrary, the orthogonal slicing performance is analyzed based on the number of allocated eMBB and uRLLC RBs at each time-slot under CES and SRS. To the best of our knowledge, no prior work exists that has developed a cost-efficient/computational-aware RAN slicing strategy for allocating radio resources, allowing for a direct comparison with our proposed CES approach. An evaluation is carried out of the proposed approach in terms of delay experienced by the uRLLC users and the observed data rates of eMBB users, by measuring the impact that changes occurring in uRLLC slice have on the eMBB slice and vice-versa.

The rest of the chapter is organized as follows. Section 3.2 provides an overview of the existing literature highlighting the recent works on RAN slicing. Section 3.3 describes the vRAN slicing model, formulation, and optimization, under the orthogonal slicing approach. It also includes a comparative study on the performance of CES to that of SRS under different scenarios enlisted in the same section. Section 3.4 discusses the system model, problem formulation, and the CES strategy under non-orthogonal slicing. Section 3.5 describes the optimization method. The performance of our approach under different scenarios is discussed in Section 3.6. Finally, section 3.7 presents the conclusion of the work.

3.2 Related Work

Network Slicing has received a great deal of attention owing to its relevance in the support of highly demanding mobile services and applications. In particular, multiplexing between eMBB and URLLC traffic in a shared radio access network (RAN) has been tackled in [39, 40, 46, 41]. Given the limited radio resources (e.g., physical resource blocks (PRBs), transmit power) in a RAN, an efficient resource allocation among eMBB and URLLC slices is crucial to satisfy the QoS requirements of the users. To facilitate the support of the slices, 5G NR standardized the techniques

of numerology [47], mini-slot-based transmission [36], and punctured scheduling [37] to be used for service multiplexing in a RAN. Taking into account these three techniques, the RAN slicing has become a multi-timescale problem.

The existing body of work can be categorized into two main lines of research. The former pertains to the **orthogonal slicing approach**, where the wireless service provider reserves a portion of bandwidth for the eMBB users, and another portion of bandwidth for the URLLC users. In this approach, which is considered for instance in [46, 48, 49], service isolation among network slices is provided. However, the allocated resources to uRLLC slice may be underutilized due to the uRLLC traffic dynamics. Yang et al. in [46] proposed an algorithm based on sample average approximation and alternating direction method of multipliers (ADMM) techniques for a two-timescale RAN slicing problem to support multicast eMBB and bursty URLLC services. Wu et al. in [48] proposed a DRL algorithm to solve a RAN slicing problem for vehicular networks. Hua et al. [49] applied DRL to design an online RAN slicing algorithm in a single timescale framework, in which the same TTI is considered for the eMBB and URLLC users.

Conversely, the latter line of research uses **non-orthogonal slicing** with punctured scheduling. This approach, which is used in [39–44], can provide an efficient use of radio resources for uRLLC users. However, punctured scheduling may degrade the performance of eMBB slice due to the potential reduction of the eMBB users' data rate.

More in detail, an example of the first approach can be found in [50] where we designed a cost-efficient slicing strategy, named CES, that minimizes the computing cost due to slicing while guaranteeing the target data rate for eMBB users and delay of uRLLC users specified in the SLA. Looking at the second approach, instead, Bairagi et al. [39] considered the network slicing problem in a downlink orthogonal frequency division multiple access (OFDMA) system by maximizing the spectral efficiency, while guaranteeing the required data rate for the eMBB users and latency for uRLLC users, based upon puncturing technique. Anand et al. [40] considered a joint eMBB/uRLLC scheduling problem for various eMBB rate loss models while the uRLLC traffic is dynamically multiplexed with the eMBB traffic through punctured scheduling. Alsenwi et al. [41] proposed a risk-sensitive punctured scheduling approach, where the radio resources used by the eMBB users can be reallocated to the uRLLC users. Also, [42] proposed Mixed numerology Mini-slot based Resource

Allocation [MiMRA] that guarantees that the loss in eMBB data rate due to the co-existing URLLC traffic is minimal. The work in [43], instead, aims to maximize the minimum expected achieved rate of eMBB users (MEAR), and fairness among them, by employing a one-to-one matching game to compute appropriate eMBB and uRLLC pairs for uRLLC resource allocation. Finally, [51] studied the resource slicing problem and formulated it as an optimization problem that aims at maximizing the eMBB data rate subject to a uRLLC reliability constraint, while accounting for the variance of the eMBB data rate to reduce the impact of immediately scheduled uRLLC traffic on the eMBB reliability.

3.3 Orthogonal Slicing

Using the orthogonal slicing approach, the service provider allocates a portion of the total available bandwidth to each slice. The service provider can also reserve a portion of bandwidth to be shared among the network slices.

3.3.1 vRAN Slices: Modeling and Optimization

We now leverage the characterization of the vRAN computational requirements given in Section 2.3, to develop a solution framework for designing and optimizing vRANs slicing. In particular, after modeling the vRAN and the slices supported therein we formulate the problem of cost-efficient slice (CES) dimensioning, which maximizes the slice profit while accounting for the CPU cost.

System Model

We focus on a gNB supporting a group of users ($\mathscr{E}$) requiring eMBB service, and a set of users ($\mathscr{U}$) demanding uRLLC service. For simplicity of notation, we consider a set of slices $\mathscr{M}$ including only a single eMBB and a single uRLLC slice, although the extension to the case of multiple eMBB and uRLLC slices is straightforward. Time is divided into Transmission Time Intervals (TTIs), denoted by $t \in \mathscr{T} = \{1, 2, \ldots, T\}$. Radio resource is divided both in the frequency domain and in the time domain, yielding F RBs, each of bandwidth B. Considering an equal power allocation, the SINR of the generic UE i at time t is given by $\gamma_{i,j,t} = \frac{P \cdot H_{i,j,t}}{B \cdot N_0}$, where P is the transmit

power of the gNB, $H_{i,j,t}$ is the channel gain of user i on RB j at time t, and N_0 is the power of additive white Gaussian noise (AWGN).

For the conventional services, such as eMBB with large transmitted packet size, the achievable data rate of UE i for RB $j \in \mathscr{F}$ at the t-th TTI can be directly estimated according to Shannon's capacity as written below in (3.1):

$$r_{e,j,t} = \Delta t \cdot B \log_2(1 + \gamma_{e,j,t}) \tag{3.1}$$

$$r_{u,j,t} = \Delta t \cdot B \Big[\log_2(1 + \gamma_{u,j,t}) - \sqrt{\frac{C_{u,j,t}}{l_{u,j,t}}} Q^{-1}(\varepsilon) \log_2 e) \Big] \tag{3.2}$$

However, for the short-sized packet transmission (ranging from 32 bytes to 200 bytes), such as uRLLC [52], the data rate falls in the finite block-length channel coding regime [53]. Therefore, the data rate are modeled as (3.2), where

- Δt is the time duration of one TTI, which set to 1 ms,
- ε is the transmission error probability,
- $Q^{-1}(\cdot)$ is the inverse of the Gaussian Q-function,
- $l_{u,t}$ represents the length of the codeword block in symbols and can be obtained based on the selected numerology for the uRLLC slice,
- $C_{u,j,t}$ is the channel dispersion, which depicts the stochastic variability of the channel relative to a deterministic channel with the same capacity, given by $C_{u,j,t} = 1 - \frac{1}{(1+\gamma_{u,j,t})^2}$.

Notice that, to guarantee $(1-\varepsilon)$ reliability for the transmission of $r_{u,j,t}$ bits per TTI towards a user, it is required to assign sufficient RBs with a large SNR. Thus, we consider that the SNR for every user on each RB never drops below 5 dB [54].

The achievable rate of an eMBB UE, $e \in \mathscr{E}$, in TTI t is thus given by:

$$r_{e,t} = \sum_{j=1}^{F} \alpha_{e,j,t} \cdot r_{e,j,t} \tag{3.3}$$

where binary variable $\alpha_{e,j,t} = 1$ indicates that the j-th RB is allocated to UE e, and $\alpha_{e,j,t} = 0$ otherwise. The achievable rate of an uRLLC UE, $u \in \mathscr{U}$, in time slot t is instead given by:

$$r_{u,t} = \sum_{j=1}^{F} \beta_{u,j,t} \cdot r_{u,j,t} \tag{3.4}$$

where binary variable $\beta_{u,j,t} = 1$ indicates that the j-th RB is allocated to UE u and $\beta_{u,j,t} = 0$, otherwise.

Next, we introduce the SLA model, which includes data rate and packet latency as performance metrics. While the former can be derived by aggregating the amount of data that is successfully transmitted over time, a queuing model of UEs' packets is needed to derive the latter.

To this end, we assume that each slice has its DL queue at the gNB, and all packets belonging to a slice share the same queue. We then model the uRLLC slice queue at the gNB as an M/M/1/K queue with service rate μ and traffic arrival rate λ [55]. As μ depends upon the scheduling process at the MAC layer, while λ corresponds to the traffic rate of the users running on top of the slice, we write:

$$\mu_{u,t} = \frac{\sum_{j=1}^{F} \beta_{u,j,t} \cdot r_{u,j,t}}{L} \tag{3.5}$$

$$\lambda = \frac{|\mathscr{U}| \cdot d_u}{L} \tag{3.6}$$

where L is the packet size of the uRLLC application, $|\mathscr{U}|$ is the number of UEs belonging to the uRLLC slice, d_u is the traffic arrival rate of uRLLC service per user, and $u \in \mathscr{U}$. The queue length at the t-th TTI can be derived as [56]

$$q_{u,t} = \frac{1-\rho_{u,t}}{1-\rho_{u,t}^{K+1}} \sum_{k=1}^{K} (k-1)\rho_{u,t}^{k-1} \tag{3.7}$$

where $\rho_{u,t} = \frac{\lambda}{\mu_{u,t}}$. Little's law can then be applied to estimate the latency experienced by uRLLC packets in the corresponding queue:

$$\delta_{u,t} = q_{u,t}/\lambda\,. \tag{3.8}$$

At the t-th TTI, the delay of a packet arriving at the i-th UE is given by the sum of service time and queuing delay,

$$D_{u,t} = W_{u,t} + \delta_{u,t} \tag{3.9}$$

where $W_{u,t} = 1/\mu_{u,t}$, is the queue service time.
We then write the average packet delay of the $|\mathscr{U}|$ uRLLC UEs at the t-th TTI as,

$$\overline{D}_t = \frac{1}{|\mathscr{U}|} \sum_{u \in \mathscr{U}} D_{u,t}\,. \tag{3.10}$$

Cost-effective slicing (CES)

We formulate the problem of ensuring a Cost-Efficient Slicing (CES) of the vRAN, and provide some details of the problem solution.

Our goal is to obtain an optimal vRAN slicing control strategy that maximises the expected long-term profit of all slices. Such profit is defined as the difference between the sum of utility of all eMBB UEs across T TTIs and the normalized cost of computing resource consumption due to the slices supported on the vRAN. Specifically, the objective is twofold: (i) when the CPU capacity is sufficient, the goal is to minimize the operational cost (in terms of CPU usage) as long as the deployed slices meet the desired performance; (ii) when there is a deficit of computing capacity to meet such performance target, the aim is resource efficiency, i.e., to maximize the data rate of eMBB services and minimize the delay experienced by uRLLC users. By taking $\alpha_{e,j,t}$ and $\beta_{u,j,t}$, indicating the RBs allocation for the eMBB and uRLLC slices, as decision variables, the CES problem formulation is given by:

$$\max_{\{\alpha\},\{\beta\}} \mathbb{E}_{t \in \mathscr{T}} \left[\sum_{e \in \mathscr{E}} U_{e,t} - \sum_{m \in \mathscr{M}} \phi_m^t \right] \tag{3.11}$$

$$\text{s.t.} \sum_{e \in \mathscr{E}} \alpha_{e,j,t} + \sum_{u \in \mathscr{U}} \beta_{u,j,t} \leq 1, \quad \forall j \in \mathscr{F}, \forall t \in \mathscr{T} \tag{3.11a}$$

$$\overline{D}_t \leq D_{max} \quad \forall t \in \mathscr{T} \tag{3.11b}$$

$$\alpha_{e,j,t} \in \{0,1\}, \beta_{u,j,t} \in \{0,1\} \quad \forall u \in \mathscr{U}, \forall e \in \mathscr{E}, j \in \mathscr{F}\,. \tag{3.11c}$$

The utility ($U_{e,t}$) of the generic eMBB user e at TTI t is given by the eMBB user data rate on that TTI, i.e.,

$$U_{e,t} = \begin{cases} 1 - \text{erf}(x^{th} - x^{o}_{e,t}) & \text{if } x^{o}_{e,t} \geq x^{th} \\ \text{erf}(x^{th} - x^{o}_{e,t}) & \text{otherwise} \end{cases} \tag{3.12}$$

where

- x^{th} is the target data rate of an eMBB UE and $x^{o}_{e,t}$ is the observed data rate of user e on TTI t. Our choice of erf function for estimating individual UEs utility is motivated by its shape, which takes 0 value at the origin, and gradually increases (decreases) and saturates to the maximum (minimum) value in the positive (negative) direction [26]. To meet SLAs, in this case, the observed data rate, it is crucial to allocate radio resources so that the observed data rate consistently meets or stays below target values (thresholds). Moreover, it is crucial to maintain the observed data rate as close as possible to the respective target, avoiding overshooting it for optimal utilization of network resources. Therefore, our selection of the utility function takes into account these essential properties.

- ϕ_m is the cost of computing resource consumption for deploying slice $m \in \mathscr{M}$, which, based on our experimental findings and model in Section 2.2, is given by

$$\phi_m = 3.9 \cdot n_m + 0.44 \cdot a_m + 30 \quad \forall m \in \mathscr{M} \tag{3.13}$$

  where n_m is the number of users served by slice m and a_m is the number of RBs allocated to the slice.

Constraint (3.11a) limits the RB resources, while (3.11b) guarantees that the average uRLLC users' packet delay will not exceed the target value D_{max} at any TTI. Constraint (3.11c) ensures binary-valued $\alpha_{e,j,t}$ and $\beta_{u,j,t}$.

5G NR, adhering to the principles of OFDMA technology, supports multiple waveform configurations, which results in scalable numerology. A numerology represents a set of parameters such as subcarrier spacing (SCS), PRB bandwidth, time-slot duration, and OFDM symbol duration. While LTE supports carrier bandwidths of up to 20 MHz with a mainly fixed OFDM numerology (15 kHz SCS), 5G NR offers scalable OFDM numerologies by scaling the basic LTE SCS by 2^{μ}, where

μ is an integer between 0 and 4. The numerology is selected independently from the frequency band, with possible SCS of 15 kHz to 240 kHz. Regardless of the numerology, the length of a radio frame and a subframe are always 10 ms and 1 ms, respectively, while the difference is represented by the number of time slots within a subframe. The coefficients in our model in 3.13, which represents the rate of change in CPU utilization as the number of RBs and users increase, are independent of numerology. Moreover, as already discussed in Sec. 3.3.1, we use the normalized cost of CPU resource consumption while designing our slicing solution. As a result, using the flexible frame structure of 5G NR, our models can significantly help design a slicing solution for 5G vRAN.

The problem formulation, along with the above constraints, results in a mixed integer quadratically constrained programming (MIQCP) problem. Moreover, the problem includes non-positive semi-definite quadratic equality constraints. To find the CES solution, we used Gurobi [57] where the non-linear functions are approximated as as piece-wise linear functions. When solving the model, the objective bounds section provides information on the best known objective value for a feasible solution (i.e., the objective value of the current incumbent), and the current objective bound provided by leaf nodes of the search tree. A new feasible solution is found, either by a MIP heuristic or by branching. When the gap between the best feasible solution and the best bound is smaller than the default MIPGap parameter (set to 10^{-4}), Gurobi produces an optimal termination status. Although a MIP problem is in general known to be an NP-complete problem, we were able to solve the model with an optimality gap that is at maximum just 0.01%.

3.3.2 CES Performance Evaluation

In this section, we demonstrate the effectiveness of our proposed network slice dimensioning method, CES, while also considering isolation guarantees.

We set the system parameters as presented in Table 3.1, and we compare the slice profit of CES to static resource slicing (SRS), where slice requests are processed without considering the CPU cost of the gNB due to slicing. Clearly, the fewer the RBs assigned to a slice, the higher the profit of the slice. Recall that two different slices are considered in our analysis, namely, eMBB and uRLLC; also, the results

are obtained considering two UEs (Figure 3.1) and four UEs (Figure 3.2) connected to a gNB for each of the slices.

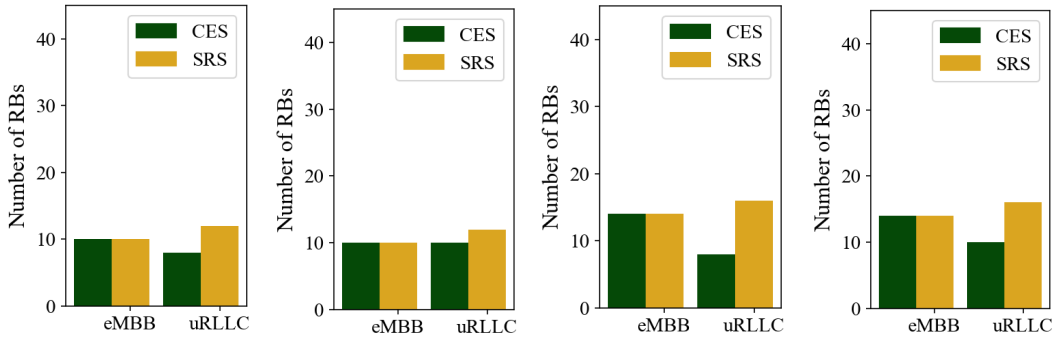
Table 3.1 Parameter settings

Parameter	Value
Number of RBs (F)/RB Bandwidth	50/180 kHz
gNB transmit gain	80 dB
D_{max}	5 ms
Service type	eMBB and uRLLC
Packet size	800 bytes (eMBB); 200 bytes (uRLLC)

Comparison of slice profit. The plots in Figure 3.1 and Figure 3.2 for the two considered scenarios present the number of RBs allocated to the slices, respectively, under our proposed scheme (CES) and under the considered benchmark (SRS).

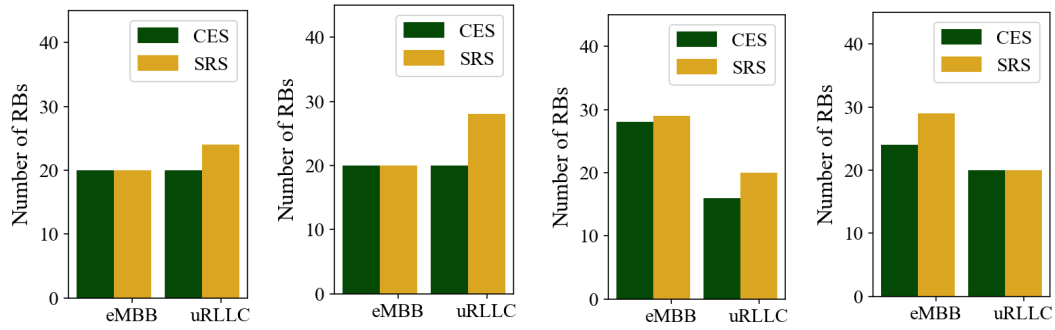
For the first scenario, in the first pair of plots (Figures 3.1(a) and 3.1(b)) and in the second pair of plots (Figures 3.1(c) and 3.1(d)), the target eMBB data rate for every UE is set to 2 Mbps and 3 Mbps, respectively, while two different values of uRLLC traffic demand are considered, namely, 0.4 packets/TTI in Figures 3.1(a) and 3.1(c), and 0.8 packets/TTI in Figures 3.1(b) and 3.1(d). The results highlight how the number of RBs allocated to the eMBB and uRLLC slices is lower under CES compared to SRS. Also, notice that the requirements in terms of delay for uRLLC traffic and data rate for eMBB traffic are always fulfilled, under both CES and SRS, as reported in Table 3.2

For the second scenario, in the first pair of plots (Figures 3.2(a) and 3.2(b)) and in the second pair of plots (Figures 3.2(c) and 3.2(d)), the target eMBB data rate for every UE is set to 2 Mbps and 3 Mbps, respectively, while two different values of uRLLC traffic demand are considered, namely, 0.1 packets/TTI in Figures 3.2(a) and 3.2(c), and 0.4 packets/TTI in Figures 3.2(b) and 3.2(d). The results highlight how the number of RBs allocated to the eMBB and uRLLC slices is less under CES compared to SRS, and it equals that of SRS only for high traffic demand of the eMBB slice. Additionally, we remark that the target delay for uRLLC traffic is always fulfilled, under both CES and SRS. The plots confirm that CES is more efficient than SRS in the support of both the uRLLC and eMBB slice: CES is able to reduce the radio resource consumption even in the presence of high eMBB traffic demand. Additionally, by looking at both Figure 3.2(d) and Table 3.3, we notice that



(a) eMBB=2Mbps uRLLC=0.40pkts/TTI (b) eMBB=2 Mbps uRLLC=0.8 pkts/TTI (c) eMBB=3 Mbps uRLLC=0.4 pkts/TTI (d) eMBB=3 Mbps uRLLC=0.8 pkts/TTI

Fig. 3.1 Comparison of the number of RBs allocated to the slices at each TTI, under CES and SRS. The traffic demand of each eMBB UE is set to 2 Mbps in (a) and (b), and to 3 Mbps in (c) and (d)



(a) eMBB=2Mbps uRLLC=0.10pkts/TTI (b) eMBB=2 Mbps uRLLC=0.4 pkts/TTI (c) eMBB=3 Mbps uRLLC=0.1 pkts/TTI (d) eMBB=3 Mbps uRLLC=0.4 pkts/TTI

Fig. 3.2 Comparison of the number of RBs allocated to the slices at each TTI, under CES and SRS. The traffic demand of each eMBB UE is set to 2 Mbps in (a) and (b), and to 3 Mbps in (c) and (d)

CES can reduce the cost due to slices support also when both the slices exhibit high traffic demand, by compromising at maximum 20% of the target eMBB data rate.

In conclusion, in all of the considered scenarios, the radio resource consumption is lower under CES than under SRS, which confirms the validity of our approach. In summary, CES performs a dual role of reducing the CPU cost and at the same time fulfilling the SLA requirements (i.e., increasing the utility of eMBB users to meet the target rate and maintaining uRLLC target delay). This strategy drives CES to efficiently allocate radio resources at the edge devices where computing resources are constrained.

Table 3.2 Achieved data rate of eMBB UEs [Mbps] and average delay for uRLLC UEs [ms] for CES and SRS, as uRLLC traffic demand [Mpbs] and target eMBB data rate [Mbps] vary for 4 connected UEs.

uRLLC traffic	x^{th}	**CES** $x^{o}_{e,t}$	**SRS** $x^{o}_{e,t}$	**CES** $\overline{D}_t$	**SRS** $\overline{D}_t$
0.4	2	2	2	0.22	0.14
0.8	2	2	2	0.20	0.16
0.4	3	2.96	2.96	0.22	0.10
0.8	3	2.96	2.96	0.43	0.17

Slice isolation. Isolation performance in network slicing can be evaluated by measuring the impact that changes (e.g., in traffic demand) occurring in certain slices have on another slice. To correctly measure and evaluate the isolation performance of the CES scheme, we consider two scenarios:

(i) Given the eMBB traffic demand, the traffic demand of the uRLLC slice is varied and the subsequent effect on the data rate of eMBB users is evaluated;

(ii) The delay experienced by uRLLC traffic is assessed, as the traffic demand of the eMBB slice varies while keeping that of the uRLLC slice fixed.

Table 3.2 and Table 3.3 illustrate the observed data rate of eMBB UEs ($x^{o}_{e,t}$) and experienced delay of uRLLC UEs ($\overline{D}_t$), for different values of uRLLC and eMBB traffic demand. Looking at the difference between the target and achieved data rate, it can be noted that the performance of the eMBB slice under CES is not affected much by the variation of uRLLC traffic and this holds also for different values of eMBB demand. Table 3.2 and Table 3.3 also present the observed uRLLC delay for different values of eMBB traffic. Importantly, the variation of the eMBB demand does not affect the delay of the uRLLC slice, which remains always below the max tolerable delay value (set to 5 ms) regardless of the value of uRLLC traffic, thus highlighting again a very good level of isolation between the two slices.

3.4 Non-orthogonal Slicing

Since the URLLC traffic has a strict latency requirement, it should be scheduled immediately upon arrival at the base station. Considering the transmission priority of the URLLC traffic in the shared bandwidth part, they can puncture (i.e., override)

Table 3.3 Achieved data rate of eMBB UEs [Mbps] and average delay for uRLLC UEs [ms] for CES and SRS, as uRLLC traffic demand [Mpbs] and target eMBB data rate [Mbps] vary for 8 UE case.

uRLLC traffic	x^{th}	**CES** $x^{o}_{e,t}$	**SRS** $x^{o}_{e,t}$	**CES** $\overline{D}_t$	**SRS** $\overline{D}_t$
0.1	1	0.96	0.96	0.25	0.13
0.4	1	0.96	0.96	0.2	0.11
0.1	2	2.0	2.0	0.25	0.19
0.4	2	2.0	2.0	0.20	0.13
0.1	3	2.8	2.96	0.36	0.25
0.4	3	2.4	2.96	0.2	0.2

some of the ongoing eMBB transmissions [58]. That is, a nonorthogonal slicing approach is used.

3.4.1 System Model and Problem Formulation

For simplicity, we start by considering a scenario with one gNB serving two user groups: $\mathscr{E}$, which requires eMBB service, and $\mathscr{U}$, which demands uRLLC service. In our simplified notation, we have a set of slices $\mathscr{S}$, consisting of a single eMBB slice and a single uRLLC slice, although the extension to multiple eMBB and uRLLC slices is straightforward. Radio resources in the frequency domain are divided into RBs $j \in \mathscr{F} = \{1,2,3,..F\}$, each with a bandwidth B determined by the numerology (μ) chosen (as shown in Table 3.4). The time domain is divided into time slots $\mathscr{T} = \{1,\ldots t\}$, each with a duration τ depending on μ. These time slots are further subdivided into mini-slots $\mathscr{M} = \{1,\ldots m\}$, with each mini-slot duration ω calculated based on the number of OFDM symbols. The arrival of uRLLC traffic at the gNB follows a Poisson distribution and occurs during any mini-slot m of a given time slot t. Each uRLLC UE $u \in \mathscr{U}$ requests a payload of size $L_u^{m,t}$ (varying from 32 to 200 bytes). gNB allots the RBs to the eMBB UEs at the commencement of any time slot $t \in \mathscr{T}$.

The achievable data rate of an uRLLC user among overlapped RBs when multiple RBs are allocated at a mini-slot m of time-slot t is given as:

$$R_u^{m,t} = \sum_{j \in \mathscr{F}} \sum_{e \in \mathscr{E}} \zeta_{e,u,j}^{m,t} \cdot r_{u,j}^{m,t}. \tag{3.14}$$

Table 3.4 Parameters of different 5G numerology settings

Numerology	0	1	2	3
Subcarrier spacing (SCS)	15 kHz	30 kHz	60 kHz	120 kHz
PRB bandwidth	180 kHz	360 kHz	720 kHz	1.44 MHz
Time slot duration	1 ms	0.5 ms	0.25 ms	0.125 ms

Table 3.5 5G numerology and the considered uRLLC transmission duration

μ	uRLLC transmission duration	Blocklength per PRB
0	2	24
1	4	48
2	8	96

where $\zeta_{e,u,j}^{m,t}$=1 indicates that $j \in \mathscr{F}$ RB of eMBB UE $e \in \mathscr{E}$ pairs with an URLLC user $u \in \mathscr{U}$ using puncturing at a mini-slot $m \in \mathscr{M}$ of time-slot $t \in \mathscr{T}$, and $\zeta_{e,u,j}^{m,t}$=0 otherwise. $r_{u,j}^{m,t}$ is the achievable rate of an RB j of an uRLLC user u. The data rate falls in the finite block length channel coding regime due to short-sized packet transmission of uRLLC and is approximated as, [53]

$$r_{u,j}^{m,t} = B_\mu \log_2(1+\gamma_{u,j}^{m,t}) - \sqrt{\frac{C_{u,j}^{m,t}}{l_{u,j}^{m,t}}} Q^{-1}(\varepsilon)\log_2 e) \tag{3.15}$$

where $l_{u,t}^{m,t}$ represents the length of the codeword block in symbols and can be obtained according to Table 3.5 based on the selected μ for the uRLLC slice. The other parameters are already detailed in our previous work [59].

For conventional services, such as eMBB with large transmitted packet size, the achievable data rate of an eMBB user e for a given RB at time slot t can be directly estimated according to Shannon's capacity as,

$$r_{e,j}^{t} = B_\mu \log_2(1+\gamma_{e,j}^{t}) \tag{3.16}$$

where $\gamma_{e,j}^{t} = \frac{P_e \cdot |h_{e,j}^{t}|^2}{N_e}$ represents the signal to noise ratio (SNR). P_e, h_e, and N_e indicate the transmission power, channel gain, and channel noise, respectively, for user $e \in \mathscr{E}$. The achievable rate of the eMBB UE, $e \in \mathscr{E}$, in Transmission Time Interval (TTI) t

Table 3.6 Summary of notations

Symbol	Meaning
$\mathscr{S}$	Set of slices
$\mathscr{E}$	Set of eMBB users
$\mathscr{U}$	Set of uRLLC users
$\mathscr{F}$	Set of RBs of uniform bandwidth B
$\mathbf{U}$	Set of numerologies
B_μ	Bandwidth of an RB in numerology μ
τ_μ	Duration of a time-slot in numerology μ
ω	Duration of a mini-slot
$\mathscr{M}$	No. of mini-slots in a time-slot
$\mathscr{T}$	Total number of time-slots
$R_u^{m,t}$	Achieved data rate of an uRLLC user at mini-slot m of time-slot t
r_e^t	PRB rate for eMBB user e
R_e^t	Achieved data rate of an eMBB user at time-slot t
R_{min}	Minimum expected achieved rate (MEAR) among all eMBB users
U	Utility function for an eMBB user
ϕ_s	CPU cost function for deploying slice s
λ	arrival rate of uRLLC traffic in a mini-slot m of time-slot t
D_{max}	Maximum tolerable uRLLC delay
x_{th}	Target data rate of an eMBB user
α	Resource allocation vector for an eMBB user
ζ	Resource allocation vector for punctured eMBB and uRLLC pairs
C	Constant number of RBs
γ_e^t	SNR of eMBB user e in time-slot t
$\gamma_u^{m,t}$	SNR for uRLLC user u from gNB at mini-slot m of time-slot t

is given by:

$$R_e^t = \left\{ \sum_{j=1}^{F} \alpha_{e,j}^t - \sum_{j=1}^{F} \sum_{u \in \mathscr{U}} \sum_{m \in \mathscr{M}} \zeta_{e,u,j}^{m,t} \right\} \cdot r_{e,j}^t \tag{3.17}$$

where binary variable $\alpha_{e,j}^t = 1$ indicates that the j-th RB is allocated to UE e at TTI t, and $\alpha_{e,j}^t = 0$ otherwise,

The average achievable data rate for the eMBB user $e \in \mathscr{E}$ is then given by,

$$\overline{R}_e = \frac{1}{|T|} \sum_{t=1}^{|T|} R_e^t . \tag{3.18}$$

Importantly, the eMBB data rate loss is associated with the uRLLC overlapping technique such as puncturing. Thus, eMBB users that lose their resources by sharing their allocated resources with uRLLC users should be guaranteed a more significant proportion of resources in the long run. We therefore consider as primary performance metric for eMBB users the Minimum Expected Achieved Rate (MEAR) [43, 39], i.e.,

$$R_{\min} = \min_{e \in \mathscr{E}} (\overline{R}_e) \,. \tag{3.19}$$

Next, we introduce the SLA model, which includes both data rate and packet latency as performance metrics as discussed in Section 3.3.1. As σ depends upon the scheduling process at the MAC layer, while λ corresponds to the traffic rate of the users running on top of the slice, we write:

$$\sigma_{u,m,t} = \frac{\sum_{j \in \mathscr{F}} \zeta_{e,u,j}^{m,t} \cdot R_u^{m,t}}{L} \tag{3.20}$$

$$\lambda = \frac{|\mathscr{U}| \cdot d_{u,m,t}}{L} \tag{3.21}$$

$|\mathscr{U}|$ is the number of UEs belonging to the uRLLC slice, $d_{u,m,t}$ is the traffic arrival rate of uRLLC service per user in each mini-slot m of time-slot t. The queue length at each mini-slot m of time-slot t can be derived as [56]

$$q_{u,m,t} = \frac{1 - \rho_{u,m,t}}{1 - \rho_{u,m,t}^{K+1}} \sum_{k=1}^{K} (k-1) \rho_{u,m,t}^{k-1} \tag{3.22}$$

where $\rho_{u,m,t} = \frac{\lambda}{\sigma_{u,m,t}}$. Little's law can then be applied to estimate the latency experienced by uRLLC packets in the corresponding queue:

$$\delta_{u,m,t} = \frac{q_{u,m,t}}{\lambda} \,. \tag{3.23}$$

At mini-slot m of time-slot t, the delay of a packet arriving at the u-th UE is given by the sum of service time and queuing delay,

$$D_{u,m,t} = W_{u,m,t} + \delta_{u,m,t} \tag{3.24}$$

where $W_{u,m,t} = \frac{1}{\sigma_{u,m,t}}$, is the queue service time.

3.4.2 The Cost-Effective Slicing (CES) Strategy

Our objective is to derive an optimal RAN slicing control strategy in 5G NR that maximizes the long-term profit of all slices. This profit is defined as the difference between the utility of eMBB UEs across t time-slots and the normalized cost associated with computing resource consumption resulting from the slices supported on the RAN.The utility of eMBB users is given by:

$$U = \begin{cases} 1 - \text{erf}(x^{th} - x^{o}) & \text{if } x^{o} \geq x^{th} \\ \text{erf}(x^{th} - x^{o}) & \text{otherwise} \end{cases} \tag{3.25}$$

where x^{th} is the target per-UE data rate for eMBB traffic and x^{o} is the observed minimum expected achieved data rate (MEAR) over all eMBB users (i.e., the observed value of R_{min}). It is essential to keep the observed data rate as close as possible to the respective target avoiding going beyond that for optimum utilization of network resources: substantially better values than the target ones would indeed translate into a waste of resources. Thus, our choice of utility function equally accounts for the aforementioned properties.

The computing resource consumption for deploying slice $s \in \mathscr{S}$, denoted with ϕ_s, is instead based on our experimental findings [30, 59] and is given by:

$$\phi_s = 3.9 \cdot n_s + 0.44 \cdot a_s + 30 \quad \forall s \in \mathscr{S} \tag{3.26}$$

where n_s is the number of users served by slice s and a_s is the number of RBs allocated to the slice.

By taking $\alpha^{t}_{e,j}$ and $\zeta^{m,t}_{e,u,j}$, indicating the RBs allocation for the eMBB and uRLLC slices (resp.), as decision variables, the CES problem formulation can then be written

as:

$$\mathbf{P_0}: \max_{\{\alpha\},\{\zeta\}} U(\{\alpha\},\{\zeta\}) - \mathbb{E}_{t\in\mathscr{T}}\left[\sum_{s\in\mathscr{S}} \phi_s^t(\{\alpha\},\{\zeta\})\right] \tag{3.27}$$

$$\text{s.t. } D_{u,m,t} \leq D_{max}, \quad \forall u \in \mathscr{U}, m \in \mathscr{M}, t \in \mathscr{T} \tag{3.27a}$$

$$\sum_{e\in\mathscr{E}} \alpha_{e,j}^t \leq 1, \quad \forall j \in \mathscr{F}, \forall t \in \mathscr{T} \tag{3.27b}$$

$$\sum_{e\in\mathscr{E}} \sum_{u\in\mathscr{U}} \zeta_{e,u,j}^{m,t} \leq 1, \quad \forall j \in \mathscr{F}, m \in \mathscr{M}, t \in \mathscr{T} \tag{3.27c}$$

$$\sum_{j\in\mathscr{F}} \sum_{e\in\mathscr{E}} \alpha_{e,j}^t \leq |\mathscr{F}|, \forall t \in \mathscr{T} \tag{3.27d}$$

$$\sum_{j\in\mathscr{F}} \sum_{e\in\mathscr{E}} \sum_{u\in\mathscr{U}} \zeta_{e,u,j}^{m,t} \leq |\mathscr{F}|, \forall m \in \mathscr{M}, t \in \mathscr{T}, \tag{3.27e}$$

$$\sum_{j\in\mathscr{F}} \zeta_{e,u,j}^{m,t} \geq 1, \forall e \in \mathscr{E}, u \in \mathscr{U}, m \in \mathscr{M}, t \in \mathscr{T} \tag{3.27f}$$

$$\alpha_{e,j}^t, \zeta_{e,u,j}^{m,t} \in \{0,1\} \quad , \forall u \in \mathscr{U}, e \in \mathscr{E}, j \in \mathscr{F}\ m \in \mathscr{M}, t \in \mathscr{T} \tag{3.27g}$$

where, for clarity, in the objective function we highlighted the dependency of the utility and of the computational cost of a slice on the number of radio resources ($\{\alpha\}$ and $\{\zeta\}$) allocated to eMBB and uRLLC users (resp.).

The uRLLC latency constraint is established in (3.27a), which guarantees that the uRLLC users' packet delay will not exceed the target value D_{max}. Constraint (3.27b) states that every RB can be allocated to at most one eMBB user, while (3.27c) ensures that every RB is used by at most one uRLLC user. The total number of resources allocated to all eMBB users in the system is constrained by (3.27d). Additionally, (3.27e) places a limit on the maximum number of RBs that can be allocated to arriving uRLLc users within a mini-slot. Constraint (3.27f) guarantees the allocation of at least one RB to a uRLLC user. Constraint (3.27g) specifies that each vector element of α, ζ is binary.

The problem formulation, along with the constraints, results in a mixed-integer quadratically constrained problem (MIQCP), which is NP-hard. It is thus essential to simplify the problem to reduce its computational complexity and make it solvable in a reasonable time in practical system scenarios.

3.5 Optimization Method

In light of the complexity of the optimization problem $\mathbf{P_0}$, we envision a lower-complexity solution strategy by leveraging the concept of divide-and-conquer [60]. We thus divide $\mathbf{P_0}$ into two sub-optimization problems and solve the new problems as set forth below:

- Subproblem 1 ($\mathbf{P_1}$) – Resource allocation for eMBB UEs on a time-slot basis
- Subproblem 2 ($\mathbf{P_2}$) – Resource allocation for uRLLC UEs on a mini-slot basis.

Subproblem 1. Given the short duration of a time slot, it is fair to assumed that eMBB UEs have a high demand for data over the whole considered slot. Consequently, at the beginning of every time slot, $t \in \mathscr{T}$, the eMBB users are allocated with RBs, and the allocated resources remain unchanged throughout the time slot. Then, by setting, in this first stage, all $\zeta_{e,u,j}^{m,t}$'s equal to zero, we formulate the first sub-problem as:

$$\mathbf{P_1}: \max_{\{\alpha\}} U(\{\alpha\}) - \mathbb{E}_{t\in\mathscr{T}}\left[\sum_{e\in\mathscr{E}} \phi_e^t(\{\alpha\})\right] \tag{3.28}$$

$$\text{s.t.} \sum_{e\in\mathscr{E}} \alpha_{e,j}^t \leq 1, \quad \forall j \in \mathscr{F}, \forall t \in \mathscr{T} \tag{3.28a}$$

$$\sum_{j\in\mathscr{F}}\sum_{e\in\mathscr{E}} \alpha_{e,j}^t \leq |\mathscr{F}|, \quad \forall t \in \mathscr{T} \tag{3.28b}$$

$$\alpha_{e,j}^t \in \{0,1\}, \quad \forall e \in \mathscr{E}, j \in \mathscr{F}\, t \in \mathscr{T}. \tag{3.28c}$$

Subproblem 2. When uRLLC traffic requests arrive during any mini-slot m of time slot t, the scheduler aims to fulfill these requests in the subsequent mini-slot $(m+1)$. The task involves evaluating suitable eMBB users to pair with the set of arrived uRLLC users while maintaining fairness among eMBB users. We then set in $\mathbf{P_0}$ all $\alpha_{e,j}^t$'s to the values obtained by solving $\mathbf{P_1}$, and we formulate the second

sub-problem as follows:

$$\mathbf{P_2}: \max_{\{\zeta\}} U(\{\zeta\}) - \mathbb{E}_{t\in\mathscr{T}}\left[\sum_{u\in\mathscr{U}} \phi_u^t(\{\zeta\})\right] \tag{3.29}$$

$$\text{s.t. } D_{u,m,t} \leq D_{max}, \quad \forall u \in \mathscr{U}, m \in \mathscr{M}, t \in \mathscr{T} \tag{3.29a}$$

$$\sum_{e\in\mathscr{E}}\sum_{u\in\mathscr{U}} \zeta_{e,u,j}^{m,t} \leq 1, \quad \forall j \in \mathscr{F}, m \in \mathscr{M}, t \in \mathscr{T} \tag{3.29b}$$

$$\sum_{j\in\mathscr{F}}\sum_{e\in\mathscr{E}}\sum_{u\in\mathscr{U}} \zeta_{e,u,j}^{m,t} \leq |\mathscr{F}|, \quad \forall m \in \mathscr{M}, t \in \mathscr{T}. \tag{3.29c}$$

To further clarify the above solution approach, let us refer to the following simple example. Consider that, at the beginning of time slot $t-1$, there are 3 eMBB UEs and each is assigned 4 RBs. Within $t-1$, a service request for uRLLC UEs arrives and the necessary RBs are allocated as overlapped uRLLC traffic in the mini-slots. For instance, during this time, 4, 7, and 2 RBs of eMBB UEs 1, 2, and 3 are allocated to uRLLC UEs, respectively. Therefore, the data rate of eMBB UEs 1, 2, and 3 drops by 4 RBs·1 mini-slot, 7 RBs·1 mini-slot, and 2 RBs·1 mini-slot, respectively. At the start of the next time slot, t, the gNB acknowledges the resource scheduling of uRLLC UEs in time slot $t-1$ to compensate eMBB UE 1, 2, and 3 for their reduced data rate. In particular, the gNB will allocate more RBs to such eMBB users in a fair manner, that is, with, e.g., UE 2 receiving a higher number of additional allocated RBs than 3.

3.5.1 Low-Complexity Heuristic for Sub-Problem $\mathbf{P_1}$

To ensure a fair share of resources among the eMBB users, resource allocation at a given time slot t has to account for the data rate such users experienced in the previous time slot $(t-1)$. As $\mathbf{P_1}$ (3.28) is still an NP-hard problem, a low-complexity resource allocation algorithm has to be used. To this end, we draw on the solution proposed in [39, 43] and enhance it to adapt it to our specific problem. The algorithm we apply consists of the following steps:

1. Initialization: A fixed number of RBs, N, are initially allocated to every eMBB user $e \in \mathscr{E}$, so that the target eMBB data rate is fulfilled.

2. At the beginning of slot $t \in T$, evaluate previously achieved data rates of all eMBB users. That is, get $R_e^{t-1}, \forall e \in \mathscr{E}$ from eMBB-uRLLC pairing and uRLLC resource allocation by solving $\mathbf{P_2}$.

3. For each RB $j \in \mathscr{J}$, with $|\mathscr{J}| = |\mathscr{F}| - N \cdot |\mathscr{E}|$, compute the rationality factor for every eMBB user e, defined as

$$H(e) = \frac{R_e^t + R_e^{t-1}}{\overline{R}^{t-1}}. \tag{3.30}$$

4. Assign RB $j \in \mathscr{J}$ to the user with the least value of $E(e)$.

5. Repeat step 3 to 4 for all the available RBs in $\mathscr{J}$.

Algorithm 1 Heuristic Algorithm for Solving $\mathbf{P_1}$

Initialize:
  $i = 1$, $E(e) \leftarrow N, \forall e \in \mathscr{E}$
while $i \leq |\mathscr{T}|$ **do**
  Get $R_e^{t-1}, \forall e \in \mathscr{E}$ from eMBB-uRLLC pairing and uRLLC resource allocation (Solve $\mathbf{P_2}$).
  $remRB \leftarrow |\mathscr{F}| - N \cdot |\mathscr{E}|$
  for $j = 1 : remRB$ **do**
    for $k = 1 : |\mathscr{E}|$ **do**
      $H(k) \leftarrow \frac{nRB \cdot r_k^t + R_e^{t-1}}{\overline{R}^{t-1}}$
    end for
    $idx \leftarrow \{e : e = \text{argmin}_{\mathscr{E}} H(e)\}$
    $E(idx) \leftarrow E(idx) + 1$
  end for
  $i \leftarrow i + 1$
end while
Determine $R_e^t, \forall e \in \mathscr{E}$
Determine $\mathbb{E}\left[\sum_{t \in \mathscr{T}} R_e^t\right], \forall e \in \mathscr{E}$

To summarize, at t=1, the algorithm allocates resources equally (i.e., N RBs to each eMBB user). Then, it allocates resources to eMBB UEs in the rest of the time slots depending on the previous time slot. More specifically, it considers the rationality $H(e)$, which is the fraction of the sum of achieved data rate of a given eMBB user involving the current time-slot t ($R_e^t = N \cdot r_e^t$) and the previous time

slot $(t-1)$ (R_e^{t-1}) relative to the average achieved data rate across all eMBB users ($\overline{R}^{t-1}$). A low achieved eMBB data rate in the previous time slot results in a lower rationality for a particular eMBB user. Thus, the eMBB user with the least achieved data rate due to uRLLC puncturing of eMBB RBs or weak channel conditions in time slot $t-1$ has higher priority to be allocated the RB. In this way, the algorithm can accommodate the MEAR of eMBB UEs in the long run adequately and in a fair manner.

Complexity analysis. For each time slot, let N be the number of RBs assigned initially to all eMBB UE where $N \leq |\mathscr{F}|$. The remaining number of RBs to be allocated to the most suffering eMBB users is $(|\mathscr{F}| - N)$. The complexity required for each RB allocation is $O(|\mathscr{E}|)$. The eMBB resource allocation in each time slot takes $((|\mathscr{F}| - N)|\mathscr{E}|)$. It follows that the overall complexity is $O(|\mathscr{T}|\mathscr{F}||\mathscr{E}|)$.

3.5.2 Solving Sub-Problem $\mathbf{P_2}$

We reformulate the second sub-problem (3.29) to take into account the CPU cost associated with both the eMBB and the uRLLC slice. Thus, we write $\mathbf{P_2}$ as:

$$\max_{\{\zeta\}} \sum_{e \in \mathscr{E}} U(\{\zeta\}) - \mathbb{E}_{t \in \mathscr{T}} \left[\sum_{s \in \mathscr{S}} \phi_s^t(\{\zeta\}) \right]. \tag{3.31}$$

In contrast to the definition of U in Eq. (3.25), x^o here is the average achievable data rate of an eMBB user e in time-slot t and is given by Eq. (3.17).

Complexity analysis. The problem formulation (3.31), along with the constraints (3.29a–3.29c), results in an MIQCP problem, which can be solved using Gurobi [57]. To solve the model, the non-linear functions (objective function and quadratic constraints) are approximated as piece-wise linear functions. Then, a feasible solution is found, either by a MIP heuristic or by branching. When the gap between the best feasible solution and the best bound is smaller than the default MIPGap parameter (set to 10^{-4}), it is considered that the optimal solution has been attained.

3.6 Numerical Analysis

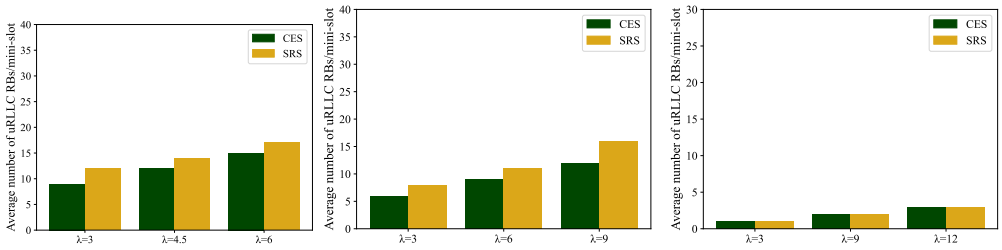
In this section, we first describe the scenario we use for our performance evaluation. Then we show the performance of our proposed approach, CES, through an extensive experimental analysis, and compare it against the Static Resource Slicing (SRS) approach [45, 42] where slice requests are processed without considering the CPU cost of the gNB due to slicing. As mentioned, SRS has been selected as benchmark, since, to the best of our knowledge, no existing scheme for radio resource allocation accounts for cost-efficient/computational-aware RAN slicing.

Table 3.7 Simulation parameters

Parameter	Value
Total channel bandwidth	20 MHz
Carrier frequency	2.62 GHz
Maximum BS transmission power	24 dBm
BS coverage radius	500 m
Noise spectral density	-114 dBm
Channel Model	FSPL
Numerology(μ)	{0, 1, 2}
Mini-slot duration	0.25 ms
Target uRLLC delay	1 ms
uRLLC packet size (bytes)	32
Number of UEs	6 (eMBB (3), uRLLC (3))

3.6.1 Reference Scenario

In our study, we consider a shared 5G NR infrastructure with coexisting uRLLC and eMBB users. We consider one gNB operating in the Frequency Range (FR)-1, with a maximum transmission power of 24 dBm and covering a radius of 500 m. The transmission occurs at the 2.5 GHz frequency band with a total channel bandwidth of 20 MHz. The arrival of uRLLC traffic at mini-slot m of time-slot t follows a Poisson distribution with mean λ, and the uRLLC packet size is set to 32 bytes. We adopt a full buffer model for eMBB buffers at the base station, assuming a continuous data flow. The gNB utilizes numerology $\mu = 0, 1, 2$ to transmit eMBB and uRLLC traffic over all of the available RBs in each numerology. The corresponding time slots for each numerology, $t_{\mu=0} = 1$ ms, $t_{\mu=1} = 0.5$ ms, and $t_{\mu=1} = 0.25$, are sub-divided into



(a) μ=0 (b) μ=1 (c) μ=2

Fig. 3.3 Comparison of the number of RBs allocated to the uRLLC slice at each mini-slot of a time-slot, under CES and SRS for different uRLLC traffic demands (λ). The traffic demand of each eMBB UE is set to 4 Mbps and the numerology (μ) considered is 0, 1, 2 in (a) (b), and (c), respectively.

a number of M_0, M_1, and M_2 mini-slots, respectively. The mini-slot duration (ω) is 250 μs, which is sufficient to meet the latency requirement for uRLLC traffic, and it is kept the same for all considered numerologies. Additionally, the simulation incorporates a maximum tolerable delay of 1 ms, with the consideration that eMBB traffic is not as time-sensitive as uRLLC traffic.

3.6.2 CES Performance Evaluation

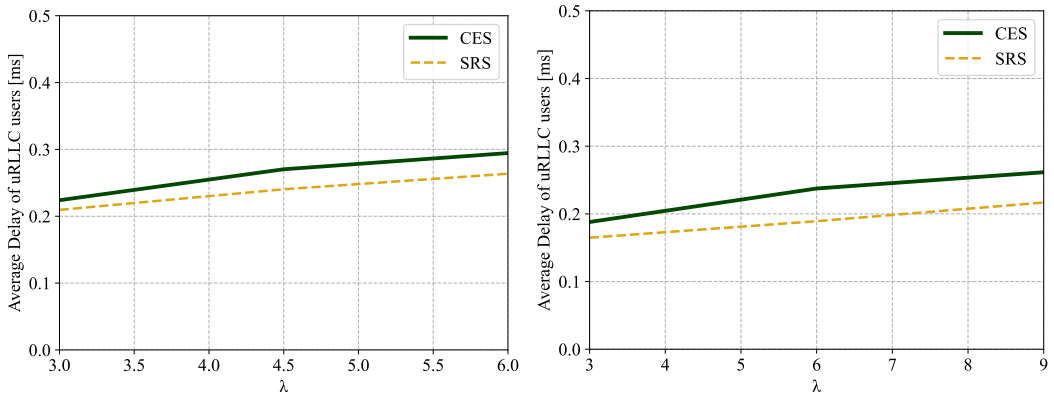
We showcase the effectiveness of our proposed slice-dimensioning method, CES, taking into account the performance requirements of the eMBB and uRLLC slices. We configure the system parameters as outlined in Table 3.7, and we compare the slice profit of CES to static resource slicing (SRS), as shown in Fig. 3.3, where minimizing the number of RBs assigned to a slice leads to higher slice profit. In SRS, slice requests are processed without considering the CPU cost of the gNB due to slicing. The objective of the SRS scheduler (similar to the Sum-Rate [45] scheduler, MiMRA [42]) is to maximize the average sum rate of eMBB users using the puncturing strategy. In our analysis, we consider two distinct slices namely, eMBB and uRLLC.

Comparison of slice profit. Fig. 3.3 represents the number of RBs allocated to the uRLLC slice every mini-slot, under our proposed scheme (CES) and under the considered benchmark (SRS). The target data rate of every eMBB user is set to 4 Mbps and the traffic demand (λ) of every uRLLC user is varied. The results depict that the number of RBs allocated to uRLLC users in every mini-slot is always

lower under CES compared to SRS for every uRLLC traffic demand in Numerology 0 (3.3a) and 1 (3.3b), while it is the same for Numerology 2 (3.3c). CES indeed maximizes the slice profit by allocating a lower number of RBs than SRS while satisfying the SLAs: the higher the number of RBs allocated to a slice, the higher the CPU cost/utilization of the RAN due to slicing of the radio resources, and the lower the slice profit.

To further illustrate the comparison based on the numerology schemes, it is worth mentioning that in higher numerology schemes (e.g., μ=1 and μ=2) the number of allocated uRLLC RBs is noticeably less compared to lower numerology scheme (μ=0). This reduction is due to the higher PRB rate, scaled up by a factor of 2^{μ}, in the higher numerology schemes. For instance, when the traffic demand λ is set to 3, the allocated RBs in μ=1 and μ=2 are significantly fewer than those in μ=0. Building on our earlier discussion regarding our proposed cost-efficient scheme (CES), it becomes evident that the impact on CPU cost/consumption increases with the rising number of required RBs. In the case of Numerology 2 (3.3c), where the required RBs are fewer, CES experiences a reduced impact on CPU cost, ultimately resulting in the number of allocated RBs being equivalent to that of SRS. To showcase/demonstrate the effectiveness of our proposed approach, specifically in terms of the number of allocated RBs, in Fig. 3.3b and Fig. 3.3c, we deliberately select higher values of traffic demand (λ). We remark that we evaluated the performance of CES only in terms of the number of allocated radio resources, since, as it can be noted in our earlier work [30, 59], the dominant impact on the CPU consumption is represented by the number of connected UEs, rather than by the number of allocated RBs. In addition, we would like to highlight that further considerations about the CPU consumption can be made starting from the results in Fig. 3.3a and Fig. 3.3b, which show how CES allocates a lower number of RBs, with respect to SRS. The smaller the number of radio resources allocated, the lower the CPU utilization of the virtual gNB according to Eq. 3 in [59].

eMBB and uRLLC Slice Performance. The performance of slices can be effectively evaluated by gauging the influence that alterations, such as shifts in traffic demand, in one slice exert on another. In our assessment of the proposed approach CES, we focused on measuring performance in terms of the delay encountered by uRLLC users and the observed data rates of eMBB users. This evaluation was conducted by varying the uRLLC traffic demand and adjusting the target data rate of eMBB users.



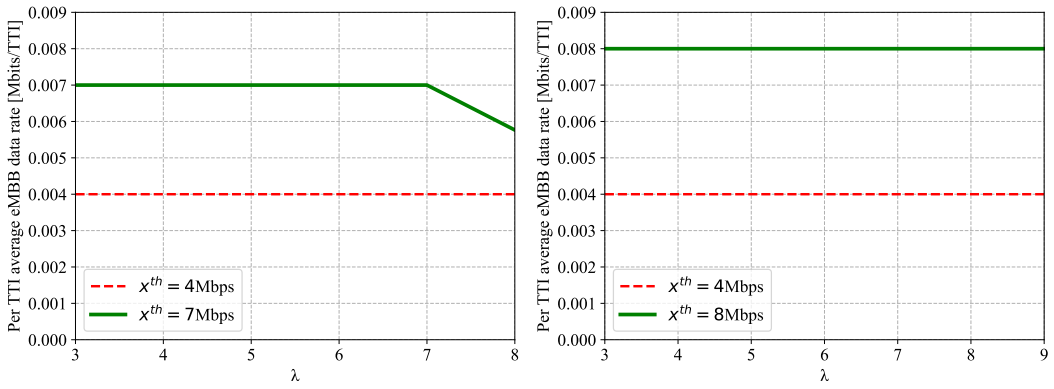
(a) Numerology μ=0 (b) Numerology μ=1

Fig. 3.4 Average delay for uRLLC UEs [ms] for CES and SRS, as uRLLC traffic demand (λ) is varied for Numerology μ=0 in (a) and μ=1 in (b).

Initially, we assess the delay experienced by uRLLC users under different uRLLC traffic demands. For the first and second scenarios, illustrated in Figures ì3.4a and 3.4b the numerology considered is 0 and 1, respectively. In these scenarios, the target eMBB data rate for each UE is fixed to 4 Mbps while the uRLLC traffic demand is varied for all the users. The results underline that, as the uRLLC traffic demand increases, the observed delay for different values of λ always remains below the maximum tolerable delay value (set to 1 ms). However, the uRLLC delays in CES are higher compared to SRS due to the higher number of RBs allocated under SRS than under CES. An important result thus follows: at the cost of a slight increase in delay without overshooting the maximum tolerable delay, CES allows for a considerable reduction in the overall CPU consumption of the RAN compared to its benchmark.

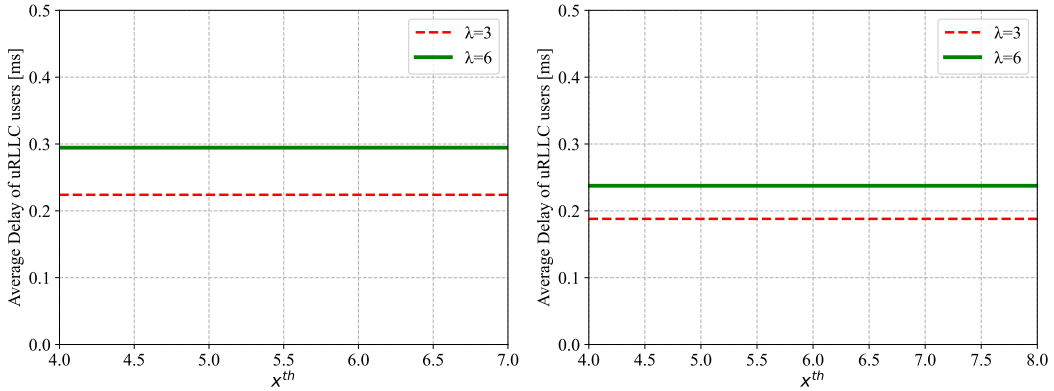
In the subsequent analysis, we vary the traffic demand of the uRLLC slice while maintaining a constant eMBB traffic demand, hence eMBB target performance. The impact on the achieved data rate of eMBB users is then evaluated for our proposed approach CES. Fig. 3.5 illustrates the average observed data rate of the eMBB users with respect to the uRLLC traffic load for two numerologies (namely, 0 and 1). We set x^{th}=4 Mbps and 7 Mbps, respectively, as the two target data rates for each eMBB user in μ=0 (see Fig. 3.5a).

Subsequently, we analyze the performance of CES in managing incoming uRLLC load. As observed in Fig. 3.5a, the eMBB users consistently maintain their target data rate when uRLLC traffic demand (λ) is varied from 3 to 7. However, when



(a) Numerology μ=0 (b) Numerology μ=1

Fig. 3.5 Per-TTI average data rate of eMBB users with two target data rates of 4 and 7 Mbps, respectively, in Fig. 3.5a, and 4 and 8 Mbps, respectively, in Fig. 3.5b. The traffic demand of uRLLC users (λ) is varied in both cases.



(a) Numerology μ=0 (b) Numerology μ=1

Fig. 3.6 Average delay for uRLLC UEs [ms], as the eMBB target performance (x^{th}) [Mbps] is varied with $\lambda = 3$ and 6, respectively, for Numerology μ=0 in (a), and with $\lambda = 3$ and 6, respectively, for Numerology μ=1 in (b).

the incoming uRLLC traffic demand goes beyond λ=7, the gNB adopts a strategy of puncturing eMBB users to prioritize serving the uRLLC traffic. In this scenario, our proposed approach CES strives to balance the needs of uRLLC traffic users while minimizing the impact on eMBB users. Consequently, the average achieved data rate of eMBB users is slightly below the target (e.g., achieved eMBB data rate around 5.8 Mbps for λ=8). The achieved data rate may vary based on the considered number of TTIs for calculating the achievable data rate per second. This outcome underscores CES's capability to either maintain the target data rate or keep

it marginally below the target as uRLLC traffic demand rises. In the case of μ=1 (3.5b), we set a higher target data for each eMBB user equal to 8 Mbps (due to the higher PRB rate in μ=1). Notably, CES consistently maintains the target data rate even when faced with increasing traffic demand for each uRLLC user. We remark that a lower number of RBs allocated to uRLLC users (as in Fig. 3.3b) prevents the gNB from puncturing RBs from eMBB users to make room for uRLLC traffic. This strategic allocation ensures that the gNB fulfills the requirements of uRLLC traffic users without compromising the resources allocated to eMBB users.

In our final evaluation, we scrutinize the delay experienced by uRLLC users while varying the traffic demand of the eMBB slice, with the uRLLC slice demand held constant. Fig. 3.6 illustrates the average delay of uRLLC users as the eMBB traffic demand is varied. The scenarios consider constant uRLLC traffic demand of λ=3 and 6 for Numerology μ=0 in (a) and Numerology μ=1 in (b), respectively. Remarkably, the observed delay consistently remains below the maximum tolerable delay value of 1 ms, and it remains constant even with higher eMBB rates. It is worth highlighting that, by puncturing the necessary number of RBs from eMBB users, CES provides the uRLLC users with the necessary RBs to meet their delay requirements while preserving the target eMBB data rate. Also, the delay is consistently lower in higher numerology schemes (Fig. 3.6b) due to the higher PRB rate in μ=1, and, as expected, the delay increases as λ grows.

3.7 Conclusion

In this paper, we addressed the cost-efficient resource allocation problem in 5G networks featuring network slicing. We formulated a resource allocation problem that maximizes the slice profit while guaranteeing uRLLC constraints with respect to latency as well as the target data rate of eMBB users. Given the problem inherent complexity, we introduced a strategic approach by decoupling the original problem into two sub-optimization problems, eMBB resource allocation and uRLLC resource allocation. We then used a simple, low-complexity heuristic for the eMBB resource allocation that maximizes the MEAR among eMBB users at time-slot boundaries. Meanwhile, for uRLLC resource allocation at every mini-slot of a time slot, we maximized the slice profit while meeting slice-specific SLAs. Our numerical results demonstrate that our approach achieves cost-efficient resource slicing, and meets the

data rate and delay requirements outlined in the SLAs for both eMBB and uRLLC slices.

Chapter 4

Fair and Scalable Orchestration of Virtual Resources for Edge Services

The convergence of service virtualization and edge computing offers the advantage of low-latency services while maintaining localized data storage and processing. However, the limited resource availability at the edge introduces a conflict when supporting both virtualized user applications and network functions. Additionally, the intertwined resource requests from user applications and network functions further complicate the scenario due to data transfer dependencies. In this work, we initially demonstrate, through experimental tests, the correlation between a video-based application and a vRAN Recognizing the intricate dynamics involved, we introduce a scalable reinforcement learning-based framework, named VERA, for resource orchestration at the edge. Leveraging Pareto analysis, VERA ensures provable fair and efficient decisions. We validate VERA through a real-time proof-of-concept implementation which is also used obtain datasets reporting real-world operational conditions and performance. Utilizing these experimental datasets, we numerically illustrate that VERA consistently meets KPI targets for over 96% of the observation period. Furthermore, our real-time implementation demonstrates similar performance, with KPI differences below 12.4%. Notably, VERA's scaling cost is 54% lower than a centralized framework based on deep-Q networks.

4.1 Introduction

NFV and edge computing are reshaping mobile services by allowing collaboration between mobile operators and third parties. This collaboration enables the delivery of diverse services, such as video streaming, gaming, virtual reality, safety applications for connected vehicles, and IoT, with significantly reduced latency and bandwidth consumption. Implemented through virtual machines or containers co-located with base stations, these services benefit from low latency and jitter, as well as local data storage and processing.

The integration of NFV, edge computing, and an efficient radio interface like O-RAN [61] is a potent approach for delivering high-quality mobile services. However, some critical aspects have been overlooked. Notably, virtualization extends beyond user applications; network services, including data radio transmission and reception, are now virtualized through Virtual Network Functions (VNFs) [62–66]. Both user-centric and network-centric virtual services can be highly computationally intensive. On the flip side, computational resources at the network edge are inherently limited [67]. Consequently, in the edge ecosystem, there's a competition for resources between user applications and network services. Therefore, the design of automated and efficient resource orchestration mechanisms becomes crucial, particularly in scenarios of resource scarcity.

Examining the computational demands of virtualized user applications and network service VNFs reveals a dependency on the volume of data each service processes, and they are intricately entangled [68]. Consider, for instance, a user application at the edge and (de-)modulation and (de-)coding functions in a virtualized radio access network (vRAN). In the case of downlink traffic, the application bitrate determines the data processed by the vRAN, while for uplink traffic, the vRAN's processed data serves as the input to the application service. However, a negative correlation can also emerge. Increased data compression by a user application elevates its computational demand but reduces the transmitted data volume, consequently decreasing the computing resources needed by the vRAN. In essence, **a correlation exists between the data processed/generated by virtual applications at the edge and network services VNFs, with this correlation being positive or negative depending on the type of involved VNFs.**

Related joint resource problems have been addressed [68] , but they often overlooked the intricate relationships among system parameters and context variables. These endeavors relied on simplifying assumptions that prove impractical in real-world scenarios [17]. Our experimental analysis in Section 4.3 sheds light on these complex couplings. Notably, contextual features of the wireless link, such as signal-to-noise ratio (SNR), radio policies like the modulation order and coding scheme (MCS) selected by the vRAN, and the allocated computing resources to the vRAN exhibit non-linear effects on latency and, consequently, impact the buffering requirements for a video-based service. **The challenges outlined above hinder the application of traditional modeling techniques for optimal resource allocation in practical situations.**

In response to these trends and challenges, we introduce VERA (Virtualized Edge for Radio and user Applications), a flexible and scalable framework utilizing a model-free reinforcement learning (RL) approach. Similar model-free approaches have been employed by other authors to tackle the intricate relationships within different aspects of a vRAN [69, 70]. However, extending such approaches to a multi-service scenario presents significant scalability issues. To address this scalability challenge, we adopt a distributed multi-agent learning approach. Crafting a multi-agent learning framework where individual agents' actions collectively satisfy the stringent capacity constraints of mobile edge platforms is inherently challenging. Drawing inspiration from [71] and other literature on autonomous driving, we decompose the policy function into two stages. The first stage generates *greedy* actions based on the environment's context, while the second refines these actions to enforce hard constraints. Notably, our use case necessitates a fair enforcement of constraints, leading us to design a novel Pareto component that ensures a fair Pareto-efficient solution.

In summary, our contributions are as follows:

- First, experimental evidence supporting our observations using a containerized edge and a software-defined-radio (SDR)-based vRAN testbed is presented.
- Then, given the intricate interplay of factors, we introduce an RL model for the effective, joint allocation of computing resources for user applications and vRAN at the edge.

- To address scalability, we employ distributed learning agents, complemented by a Pareto analysis for fair and efficient decision-making when resource utilization is constrained.
- We demonstrate the excellent performance of the VERA framework in terms of convergence and its ability to closely meet the target KPIs for all services in resource-constrained scenarios. Specifically, VERA achieves KPI targets for over 96% of the observation period post-convergence. Additionally, its performance remains consistent in our real-time proof-of-concept implementation, with KPI differences below 12.4%. Notably, VERA's scaling cost is 54% lower compared to a competitive centralized framework using deep Q networks.
- Finally, we validate our approach by implementing VERA on our testbed, confirming its preserved performance when interacting with a real system in real-time.

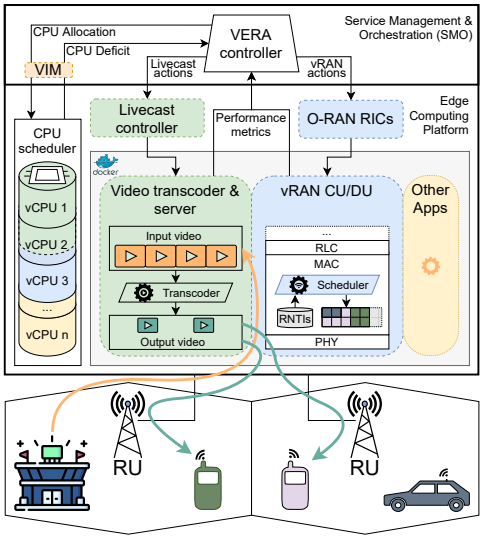


Fig. 4.1 Virtualized user application and vRAN at the edge: system scenario and reference use case

4.2 Reference Scenario and System Architecture

The system architecture and reference scenario under study (depicted in Fig.,4.1) focus on a livecast service and one virtual base station (vBS) within an edge com-

puting platform. VERA, explained in Sec.,4.4, is scalable for multiple coexisting services and vBSs/network slices.

As a sample use case, we consider a livecast service scenario, such as recording an event at a stadium. The high-quality source video is processed within an edge computing platform via a standard video transcoding service. Deploying services like livecast at the edge ensures local production and consumption of video traffic, reducing network bandwidth. Leveraging a multi-access edge computing (MEC) platform and the Radio Network Information Service (RNIS) enables real-time tuning of video coding parameters based on estimated radio channel conditions through feedback-based multicast. This approach is particularly beneficial for emerging interactive services like crowdcast, augmented reality, and mobile gaming, where video streaming plays a crucial role, and strict latency constraints are best met with an edge-based architecture.

In addition to the livecast service (as well as, possibly, other user services running at the edge), the edge computing platform hosts vBS functions, central unit (CU), and/or distributed unit (DU), jointly controlled by the VERA controller. As shown in Fig.,4.1, the VERA controller is deployed in the Service Management & Orchestration (SMO) platform. It interacts with O-RAN intelligent controllers (RIC) to configure vBS functions, edge service controllers (e.g., livecast controller), and the NFV virtual infrastructure manager (VIM) for CPU scheduler configurations, aligning with O-RAN specifications [61]. In this way, VERA's workflows, involving data collection and decision-making, adhere to O-RAN's machine learning procedures [61]. Indeed, VERA continually monitors the vRAN and livecast application states, along with overall computing resource usage in the edge platform. Leveraging these observations, it computes the values of the operating parameters for both livecast and vRAN. This ensures that, given the available computing and networking resources, the computed parameters meet the KPI targets for both applications and vRAN.

4.3 Experimental Analysis

The system architecture outlined in Sec. 4.2 has been replicated on a smaller scale in our testbed for developing and testing VERA. The key components include the edge computing platform and user equipments (UEs), communicating through an

LTE vRAN implemented using the srsRAN suite [72]. The edge platform hosts two Docker containers, representing the livecast and vRAN services, utilizing the edge resource pool.

Our vRAN testbed comprises one srseNB instance (i.e the LTE vBS) and two srsUE instances representing recipients of the video content livecast by the vBS. The livecast application involves a live video streaming transcoder implemented with ffmpeg[1] and a server based on ffserver[2]. The transcoded video, using the VP9 codec and bit rate/frame rate settings from VERA, is then served to UEs with each running a player to stream, decode, and play the video. Interaction between the radio and livecast services with VERA occurs through a dedicated API, enabling dynamic setting of operating parameters and retrieval of performance measurements. VERA also interacts with the edge computing platform OS and Docker daemon to monitor and allocate computing resources to services. More testbed details are in Sec.,4.5.2.

We employ the testbed to empirically analyze trade-offs between different actions configurable in our system under varying contexts. While our analysis initially focuses on a single user, it's essential to note that VERA supports multiple users, and later sections evaluate VERA with multiple users.

We begin by defining a contextual feature that characterizes the videos delivered by the livecast service:

- **Context 1:** Video input bit rate, input frame-per-second (FPS) rate, and input resolution;

We then introduce another feature indicating the computational demand required by the livecast service.

- **Context 2:**: Video CPU throttled time. This feature provides an indication of the processing pressure associated with the requested video, impacting overall performance and action selection.

We also define three sets of actions for our livecast service, some of which re-encode each video accordingly:

[1] https://ffmpeg.org/
[2] https://trac.ffmpeg.org/wiki/ffserver

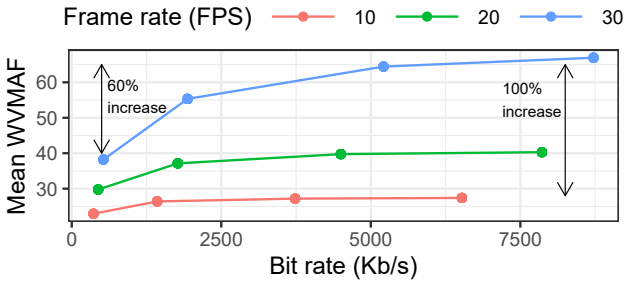


Fig. 4.2 Video quality (WVMAF) for different livecast service configurations (output FPS rate and bit rate). WVMAF increases by 100% by changing the output FPS from 10 to 30, and by 60% when increasing the output bit rate from 0.5 to 8 Mb/s

- **Action 1**: Video output bit rate.
- **Action 2**: Video output FPS rate.
- **Action 3**: CPU resources allocated to the livecast service.

and a KPI that we can use to estimate the quality of the video being delivered, as set forth below:

- **KPI 1:** Weighted Video Multimethod Assessment Fusion (WVMAF). It is derived from VMAF, an objective metric assessing video quality on a scale from 0 (worst) to 100 (best) *per video frame*. The VMAF score is computed by aggregating various components like Visual Information Fidelity, Detail Loss Metric, or Mean Co-Located Pixel Difference (details in [73]). However, VMAF assesses the quality of individual video frames, lacking the ability to measure the smoothness of a video—an aspect known to significantly impact perceived quality. WVMAF incorporates smoothness by adjusting the measured VMAF of each frame based on the output and input frame rates.

Fig. 4.2 illustrates the mean WVMAF score across various VP9-encoded videos with different output FPS rates and bit rates, CPU throttle time, and the same resolution. The results intuitively show that higher-bit-rate and higher-FPS videos yield higher WVMAF scores. It is noteworthy that the increase in mean WVMAF is 100% when the output FPS changes from 10 to 30, while it is 60% when the output bit rate increases from 0.5 to 8 Mbps. This indicates that the frame rate setting has a larger impact on WVMAF than the target bit rate, with diminishing returns for the latter. This observation is attributed to the weight amplifying the measured VMAF, influencing the score when higher frame rates are used.

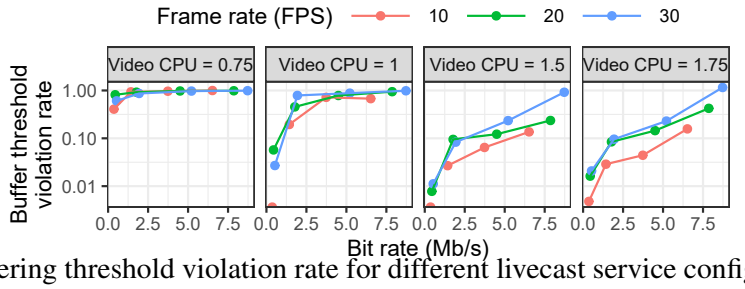


Fig. 4.3 Buffering threshold violation rate for different livecast service configurations and CPU allocations

A second relevant KPI to assess the perceived QoE of the livecast service is:

- **KPI 2:** Video player buffering. Information about the client's buffer state, which stores video frames for playout, is a good estimator of the user's QoE [74]. Specifically, when the buffer size gets close to zero, the user's player may stutter, which resorts in a low level of QoE.

In evaluating this KPI 2, we apply a threshold equivalent to 0.5 seconds of video buffered at the client's video player. Fig. 4.3 illustrates the frequency of threshold violations for the same set of videos used previously, considering different combinations of actions. Notably, the target bit rate and the allocated CPU resources, measured in units of virtual CPU (vCPU) assigned to the service, exhibit a more substantial impact on this KPI compared to the previous assessment. The ranges of this KPI span two orders of magnitude, as depicted on the logarithmic scale of the y-axis, indicating a non-linear behavior.

Next, we focus on the vRAN service, and define two more actions related to it:

- **Action 4:** CPU resources allocated to the radio; and
- **Action 5:** A Modulation and Coding Scheme (MCS) policy. This policy follows that used in [17] and imposes an upper bound on the MCS eligible by the base station, which helps to control the computational demand of the radio service;
- **Action 6:** Bandwidth, i.e., the aggregate amount of radio Resource Blocks (RBs) allocated to each UE;

and three relevant contextual features:

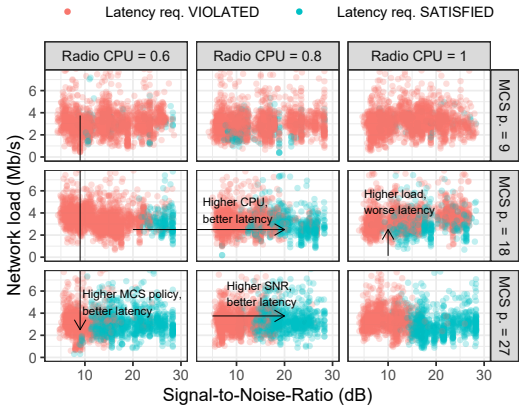


Fig. 4.4 Latency violations for different radio service configurations, contexts, and CPU allocations

- **Context 3:** Radio CPU throttled time, which is the radio counterpart of Context 2;
- **Context 4:** Signal-to-noise-ratio (SNR), which is a common feature used to estimate the quality of a wireless channel and in turn bounds its capacity; and
- **Context 5:** Network load, which corresponds to the offered load that the vBS has to process, generated by the applications deployed at the edge (such as our livecast service) and background data related to the mobile network.

Concerning radio KPIs, we first define:

- **KPI 3:** Radio latency. This is the latency associated with the data transmitted successfully over the air.

To analyze this KPI, Fig. 4.4 shows every data frame that violates/meets a latency threshold equal to 150 ms with red/blue colored dots when the vBS has to deliver randomly chosen videos from our set. We present these as functions of two of the radio contextual features (SNR and network load) and for different combinations of actions. Correlations among context, actions, and latency are evident. E.g., a higher MCS policy show a consistent improvement in latency performance, which however requires more computing resources. We observe a similar behavior when allocating a higher number of RBs (results omitted to reduce clutter). We then define one last KPI associated with the radio service:

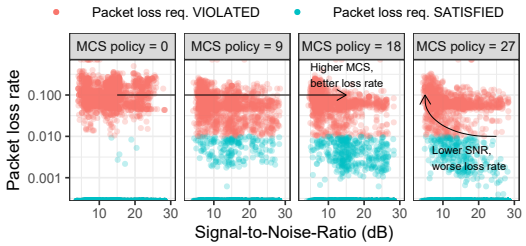


Fig. 4.5 Packet loss rate for different vRAN configurations and contexts

- **KPI 4:** Packet loss rate, which measures the number of unacknowledged TCP segments due to corruption on the radio link.

Fig. 4.5 shows this KPI for all the videos in our set as a function of the SNR (Context 4) and for different MCS policies (Action 5). In red, we mark those samples that exceed 1% packet loss. Obviously, the sample set is highly biased towards 0 packet loss rate. However, there are a number of video scenes that cause packet losses, and these are highly correlated with SNR and our MCS policy in a non-trivial manner as shown by the plot. For instance, we can observe that higher MCS policies yield considerably lower packet loss rate. Note that this gain comes from the extra wireless capacity granted by larger MCS policies and not from the reliability of a given MCS that is selected automatically by the radio scheduler: we simply impose a restriction on the set of eligible MCSs. Moreover, better SNR provides also better performance, which is intuitive. However, this relationship is non linear (note the logarithmic y axis). Likewise, the dependency between packet loss and the number of allocated RBs is also monotonic, i.e., high packet losses are observed with overly low RB allocations (like before, we omit these results to be concise).

It may be noted that we do not consider throughput, another commonly used metric, as a radio KPI in the VERA framework, since for delivering a real-time network load across a vRAN, packet loss and latency are more relevant metrics [29]. Nonetheless, it is implicit that as long as observed packet loss and observed latency values are as desired (which we later quantify in terms of KPI targets), the system throughput will be maximized since no packets are lost and all the traffic belonging to all the services is served in due time.

To conclude our experimental analysis, we plot in Fig. 4.6 a livecast KPI (buffer state) as a function of SNR (radio context) and MCS policy (radio action). Evidently, the buffer dynamics of the client's video player are highly correlated with the context and the actions performed over the livecast service. This proves that the resource

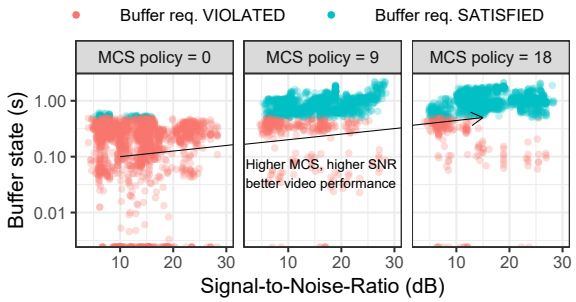


Fig. 4.6 Livecast performance (buffer state) for different vRAN configurations and contexts

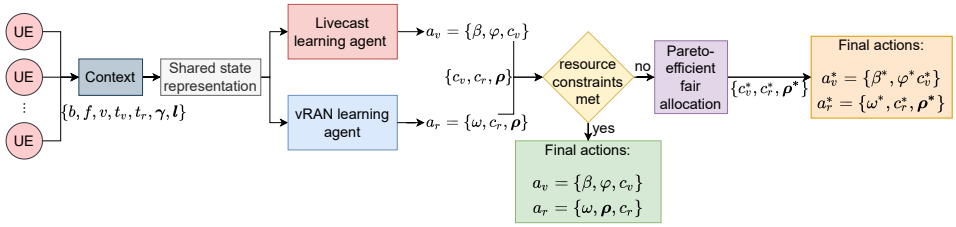


Fig. 4.7 Structure of the VERA framework

orchestration problem we endeavour into in this chapter is a coupled problem and all these edge services must be optimized jointly.

Conclusion: It is evident that the support of different applications in the edge platform leads to complex inter-dependencies between system parameters, context-action variables and KPIs, thereby making optimum resource allocation a challenging task. To this end, we propose the VERA framework, which is completely data-driven, and, hence, a good match for flexible and effective decision making in virtualized environments, despite the system complexity.

4.4 The VERA framework

The VERA framework is designed using a model-free RL approach. It includes distributed learning agents, each corresponding to a service in the edge platform, which simultaneously make decisions for the allocation of radio and computing resources as well as tune service-specific operating parameters. This design choice is key to attain *a scalable solution.* These decisions are hereafter collectively referred to as a resource allocation policy, which consists of two development stages:

- In the first stage, each RL agent makes decisions based on the shared context representation to obtain a greedy resource allocation policy;

- In the second one, greedy policies from all RL agents are collated and further refined in view of the feasibility of the chosen actions to obtain a Pareto-efficient fair resource allocation policy.

The structure of the VERA framework is shown in Fig. 4.7. Decisions are made with periodicity equal to $N \geq 1$ monitoring slots, i.e., an action is selected at the end of every decision window of duration N slots, and it is applicable to the subsequent N monitoring slots. The individual stages are elaborated in the sequel.

4.4.1 Notation

$\mathbb{R}^n$ denotes the set of n-dimensional real vectors. Vectors (usually in column form) are written in bold font, matrices are in upper-case, bold font, and sets are in calligraphic font. Subscripts and superscripts denote an element in a vector and elements in a sequence (resp.). E.g., $\langle x^{(t)} \rangle$ is a sequence of vectors with $x^{(t)} = [x_1^{(t)}, \ldots, x_n^{(t)}]^\top$ (superscript $\top$ is the transpose operator). $x_i^{(t)}$ is the i-th component of the t-th vector in the sequence.

4.4.2 Greedy analysis

Since the edge platform may have several services consuming the resource pool, owing to resource sharing, the KPI satisfaction of each is interdependent. We therefore collect input variables pertinent to different services to form a context vector. The context vector is processed through an autoencoder to create a shared context representation that captures the correlation among context variables, as well as reduces the dimensionality of the context vector. Then, each RL agent devises a greedy resource allocation policy by using the same shared context representation and by mapping it onto an action vector such that its long-term cumulative reward from the environment is maximized. Notice that, although the decisions are based on shared context, greedy policies do not ensure that the sum of capacity-constrained resources among all the services does not exceed their maximum capacity. *To solve this issue, we design a Pareto algorithm that allows for feasible and fair resource sharing.* The elements composing the greedy resource allocation policy are introduced below, while the notation we use is summarized in Tab. 4.1.

Table 4.1 Notation

Symbol	Description
Context notation	
b	Video input encoding bit rate
f	Video input FPS rate
v	Video input resolution
t_v, t_r	Normalized video and radio (resp.) throttled time
γ	CQI value
l	Livecast network load
Action notation	
β	Video output encoding bit rate
φ	Video output FPS rate
c	CPU allocation
ω	vRAN MCS policy
ρ	RB allocation
KPI notation	
ζ	WVMAF score
σ	Client's buffer state
λ	Latency
μ	Packet loss rate
$\{\zeta_o^m, \sigma_o^m, \lambda_o^m, \mu_o^m\}$	Observed KPI values by m-th UE
$\{\zeta_t, \sigma_t, \lambda_t, \mu_t\}$	Target KPI values

Context space. As described in Sec. 4.3, the resource allocation for the livecast service is governed by the following contextual information: input bit rate (b), input video FPS (f), and input resolution (v) of the streaming video (i.e., Context 1 in Sec. 4.3). Besides, to accommodate any backlog in video processing, the normalized CPU throttled time of the livecast application (t_v) in the previous monitoring slot is considered (Context 2).

Likewise, resource allocation for the vRAN is based on normalized CPU throttled time (t_r) (Context 3), the 3GPP-compliant Channel Quality Indicator (CQI) (γ) reported from UEs to vBS, which is representative of the SNR (Context 4), and the traffic from the livecast application sent over the radio link to the UEs (Context 5), specified by the network load (l). Thus, the context vector observed in monitoring slot n ($n = 1,\ldots,N$) can be written as $x^{(n)} \in \mathscr{X}$, $x^{(n)} := \{b^{(n)}, f^{(n)}, v^{(n)}, t_v{}^{(n)}, t_r{}^{(n)}, \gamma_1^{(n)}, \cdots \gamma_M^{(n)}, l_1^{(n)}, \cdots l_M^{(n)}\}$.

Further, to extract the correlation between context variables, an autoencoder projects context vector $x^{(n)} \in \mathscr{X}$ onto its latent representation $y^{(n)} \in \mathbb{R}^D$, $y^{(n)} := \{y_1{}^{(n)}, \ldots, y_D^{(n)}\}$ where $D < \dim(\mathscr{X})$. The latent representation $y^{(n)}$ is shared with each RL agent so that its decision process for a given service is informed of the performance of others accessing the resource pool, thus representing a shared context representation. The autoencoder is implemented through a simple feed forward neural network that is activated using rectified linear units in the hidden layers. Note that dimensionality reduction is only one advantage of the autoencoder: indeed, it is

primarily used to capture multimodal patterns among context variables, which may not otherwise be evident owing to the system complexity.

Action space. Since services are heterogeneous, we define action space $\mathscr{A} := \{a_k\}$, $\forall k \in (1,\ldots,K)$, comprising action vectors each having service-specific action variables. In our reference scenario, $K = 2$, and we associate $k = 1,2$, respectively, to action vectors for livecast and vRAN. Consequently, a_1 comprises the CPU allocated to the livecast application (c_v), i.e., Action 3 in Sec. 4.3, the video output encoding bitrate (β), i.e., Action 1, and the video output encoding FPS (φ), i.e., Action 2.

Conversely, a_2 includes the CPU allocated to vRAN (c_r), i.e,. Action 4, the MCS value (ω) defined before as Action 5, and the bandwidth allocated to each UE, $\rho = \{\rho_1, \rho_2, \cdots \rho_M\}$ as defined in Action 6, where M is the maximum number of users supported in the system. Here, the CPU and the radio (RB) resources are capacity constrained, i.e., $c_v + c_r \leq B_c$ and $\rho_1 + \rho_2 + \cdots \rho_M \leq B_\rho$, where B_c and B_ρ are, respectively, the total available CPU and number of RBs that can be allocated. To avoid clutter, we replace c_v and c_r with a generic c_k that denotes the CPU allocated to service k, and let ρ_m be the number of RBs allocated to UE m. Mathematically,

$$a_k = \begin{cases} (\beta, \varphi, c_k), & \text{if } k = 1, \\ (\omega, c_k, \rho), & \text{if } k = 2\,. \end{cases} \tag{4.1}$$

Next, we discretize the quantity of capacity-constrained resources that can be allocated, and map each feasible combination of action variables during the n-th monitoring slot into an action index $a_1^{(n)} := \{1,2,\ldots,N_\beta \cdot N_\varphi \cdot N_{c_v}\}$ and $a_2^{(n)} := \{1,2,\ldots,N_\omega \cdot N_{c_r} \cdot N_\rho\}$, where N_i is the number of elements in the discretized version of action variable $i = \{\beta, \varphi, \omega, c_v, c_r, \rho\}$. We remark that the action space is a mixture of variables that are inherently continuous (e.g., CPU, output bitrate, output framerate) and discrete (RBs, MCS). The primary reason for discretization of all action variables is the ease in framework implementation, since invoking different techniques for learning continuous and discrete variables will increase the system complexity by manifolds. Besides, such action definition limits the action space to a subset of discrete positive values with low cardinality, and it facilitates simultaneous selection of several resources with a single action. Instead, converting all action variables to take continuous values, and using a continuous variable-specific learning algorithm, would introduce rounding off errors in the final stage.

Reward. For a given service, KPIs are satisfied when the allocated resources make the observed KPIs to meet their respective target values. However, beside meeting the target KPIs, it is essential to keep the observed KPIs as close as possible to the target KPIs; failing that, the system may perform better than required at the cost of extra resource consumption. Consequently, the choice of a reward function should be such that it equally accounts for all the KPIs for a given service and its value increases as the observed KPIs approach the corresponding thresholds and vice versa.

Let the observed values of the livecast KPIs, i.e., WVMAF and client buffer state, and of the vRAN KPIs, i.e., latency and packet loss rate for the m-th UE, be denoted by $\zeta_o^m, \sigma_o^m, \lambda_o^m, \mu_o^m$, while the corresponding target values be $\zeta_t, \sigma_t, \lambda_t, \mu_t$, respectively. We define the reward value for m-th UE, r^m, as the sum of the reward components pertaining to each service-specific KPI k in the n-th monitoring slot within the same decision window, as:

$$r^m(y^{(n)}, a_k^{(n)}) = \begin{cases} r_\zeta^m(y^{(n)}, a_k^{(n)}) + r_\sigma^m(y^{(n)}, a_k^{(n)}), & \text{if } k = 1\,, \\ r_\lambda^m(y^{(n)}, a_k^{(n)}) + r_\mu^m(y^{(n)}, a_k^{(n)}), & \text{if } k = 2\,. \end{cases} \tag{4.2}$$

In the above expressions, $r_\zeta^m(\cdot), r_\sigma^m(\cdot)$ are the reward components from WVMAF and buffer state (resp.) for the m-th UE, given by:

$$r_{\text{KPI}}^m(y^{(n)}, a_k^{(n)}) = \begin{cases} 1 - \text{erf}(\text{KPI}_o^m(y^{(n)}, a_k^{(n)}) - \text{KPI}_t), \\ \qquad\qquad\qquad\qquad \text{if KPI is met} \\ \text{erf}(\text{KPI}_o^m(y^{(n)}, a_k^{(n)}) - \text{KPI}_t), \qquad \text{else}\,. \end{cases} \tag{4.3}$$

The terms $r_\lambda^m(\cdot)$ and $r_\mu^m(\cdot)$ are instead the reward components from latency and packet loss rate (resp.), which are given by similar expressions but with $(\text{KPI}_t - \text{KPI}_o^m(y^{(n)}, a_k^{(n)}))$ as an argument of the erf function, since all values of latency and packet loss rate lower than their respective target values are acceptable. Since the minimum and maximum values of the erf function lie between -1 and $+1$, we have: $-2 \leq r^m(y^{(n)}, a_k^{(n)}) \leq 2$. For the individual reward components, in the positive region of operation, i.e., when the KPI threshold is met, the reward value is positive and it further increases to its maximum value $+1$ as the observed KPI approaches its target KPI value. Likewise, in the negative region of operation, i.e., when the KPI threshold is not met, the value of the individual reward components is negative, which further

reduces and saturates to the minimum value -1 as the observed KPI moves away from the KPI threshold.

We recall that while devising the greedy resource allocation policy, the goal of the RL agent is to maximize the cumulative reward measured as the sum of immediate reward and future rewards over a long time horizon. To this end, we consider a generic decision window h and, extending the previous notation, we let $a_k^{(h-1)}$ denote the action for the k-th service selected in decision window $(h-1)$ and applied in decision window h. We then define the average reward over h, considering all the UEs, as

$$\bar{r}(y^{(h)}, a_k^{(h-1)}) := \frac{1}{MN} \sum_{n=1}^{N} \sum_{m=1}^{M} r^m(y^{(n)}, a_k^{(h-1)}), \tag{4.4}$$

where $y^{(h)}$ is the vector of shared contexts observed in the N monitoring slots in decision window h, while $a_k^{(h-1)}$ is the action for service k selected in decision window $h-1$ and applied in decision window h. Finally, we adopt the definition of cumulative reward for the k-th service, observed during decision window h, as the differential return $G_k^{(h)}$ defined in [75] (see Appendix A[1]).

Estimation of user buffer states. A key challenge, however, is to estimate the actual buffer dynamics *without* explicit feedback from the users. Thus it is important to design an effective *online learning* mechanism. To this end, we keep track of the time status information provided by the video encoder and the sequence number of TCP acknowledgments, all information locally available. By monitoring the amount of bytes successfully delivered, the corresponding timestamp of the scenes been transmitted, and the encoded video's frame rate, we can estimate the buffer dynamics at the client's side using simple queuing theory. Fig. 4.8 shows the evolution over time of a video player's buffer state (ground truth) vs. the inferred value for 3 trivially chosen videos and system configuration parameters. In most of the cases, our inference method is remarkably accurate. This is the case for the first two subplots in Fig. 4.8. However, we have found that there are some small number of cases where there is a non-negligible inference error. We show an example of this in the right-most plot of the figure, where we have an error of almost one second. Fortunately, these always occurs for non-critical cases, i.e., cases where the buffer state never approaches zero (the state we want to avoid). Since our estimator is

[1] https://github.com/somreetakgp/VERA-Supplementary-Material

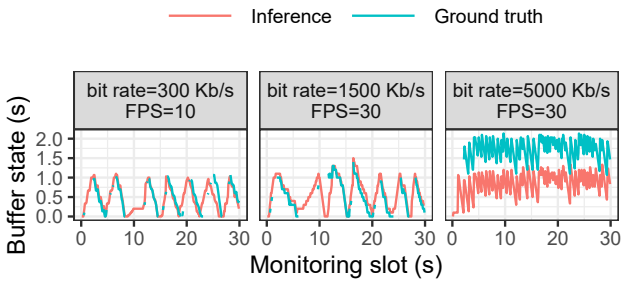


Fig. 4.8 Livecast client's buffer state inference

pessimistic, the outcome are simply more conservative decisions. We hence conclude that our inference approach is valid to compute reward.

Action-value estimation and action selection. At the end of the generic decision window h, actions need to be evaluated and the best one has to be selected. To this end, we compute the mean shared context over the N monitoring slots in h as

$$\bar{y}^{(h)} = \sum_{n=1}^{N} z_n y^{(n)} / \sum_{n=1}^{N} z_n, \tag{4.5}$$

where $z_n > 0$ and $z_N > z_{N-1} > \cdots > z_1$ are the weights assigned so that the latest shared context has the highest weight.[3] We then quantify the goodness of taking an action in response to the mean shared context using action values. For service k, the value of $a_k^{(h)}$ given policy π_k, which is $q_{\pi_k}(\bar{y}^{(h)}, a_k^{(h)})$ (see Appendix A [1])), is defined as the expected differential return conditioned on $\bar{y}^{(h)}$ and $a_k^{(h)}$, following policy π_k, i.e.,

$$q_{\pi_k}(y,a) = \mathbb{E}_{\pi_k}[G_k^{(h)} | \bar{y}^{(h)} = y, a_k^{(h)} = a]. \tag{4.6}$$

Since the context space $\mathscr{X}$ falls in the domain of real numbers, we use a practical method for action-value estimation using function approximation in an F-dimensional space, yielding the approximated function $\hat{q}_{\pi_k}(\bar{y}^{(h)}, a_k^{(h)}, w) = \sum_{f=1}^{F} w_f s_f(\bar{y}^{(h)}, a_k^{(h)})$, where $\mathbf{w}$ and $\mathbf{s}(\bar{y}^{(h)}, a_k^{(h)})$ denote the F-size weight and feature vectors (resp.), with the latter being generated using tile coding [76] (see Appendix B [1]).

The estimation of the action values is followed by an ε-greedy action selection policy [75], which selects the best action for each service so as to maximize its

[3] Although they can be arbitrarily set, we fix them to $1, \ldots, N$, in accordance with the temporal sequence of the monitoring slots.

[1] https://github.com/somreetakgp/VERA-Supplementary-Material

cumulative reward over an infinite time horizon. We consider an ε-greedy action selection with $\varepsilon = 0.5$ and ε-decay factor $= 0.999$. The ε parameter decays by a factor of 0.999 in the subsequent decision period. This favors higher exploration while the environment is still unfamiliar; with progression of time, instead, it allows for further exploitation of the environment knowledge gained during the exploration, so as to maximize the expected return.

Discussion. While single-agent RL approaches can easily solve the above capacity constraints, they suffer from the curse of dimensionality, even if implemented with deep neural networks (see DQNs) [77]. Thus, the key challenge is to provide safe (i.e., within a set of hard constraints) and fair resource allocation with a distributed multi-agent RL model that is amenable to scalable orchestration.

In this way, our distributed approach allows us to handle two sets of capacity constraints:

1. Computing resources: VERA can handle multiple vBSs (or multiple radio slices within a vBS), and multiple edge services that are competing for the same computing resource budget;

2. Radio resources: VERA can handle multiple users sharing a common carrier bandwidth.

Although for the sake of clarity the above text and Fig. 4.7 refer to two service learning agents only, one of which is a single vBS serving M UEs, the scalable multi-agent design of VERA allows for as many learning agents as services, vBSs, or slices under the above capacity constraints.

4.4.3 Pareto analysis

We recall that the CPU and RB allocation for (resp.) service k and UE m are capacity-constrained resources. Hence, it is essential that the sum of CPU (RB) allocated to different services (UEs) does not exceed the available resource budget and that the selected actions can be enacted. To this end, we introduce an algorithm that works on the multi-dimensional actions selected by the ε-greedy policy in the RL framework introduced above, and it further refines them so that the resulting actions (*i*) meet the budget constraint and (*ii*) entail fair Pareto-efficient resource sharing.

Algorithm 2 Fair Pareto-efficient Resource allocation

$S = \{\tilde{a}_k\}_k$, $\{c_k, \rho_m(k) \forall m \in \mathscr{M}\} \leftarrow \tilde{a}_k(S)$, $S' = \{c_k, \rho_m(k) \forall m \in \mathscr{M}\}, \forall k \in \mathscr{C}$ ▷ Extract capacity-constrained actions from greedy action set $\{\tilde{a}_k\}$
if $\sum_{k \in \mathscr{C}} c_k \leq B_c$ and $\sum_{m \in \mathscr{M}} \rho_m(k) \leq B_\rho, \forall k \in \mathscr{C}$ **then** ▷ Capacity-constraint check on the primary and secondary resource
    $S^\star = S'$ ▷ Output: Fair Pareto-efficient solution
else if $\sum_{k \in \mathscr{C}} c_k \leq B_c$ and $\sum_{m \in \mathscr{M}} \rho_m(k) > B_\rho$, for any $k \in \mathscr{C}$ **then** ▷ Primary resource budget constraint met, secondary resource budget constraint not met
    $\{\rho'_m(k) \forall m \in \mathscr{M}\} \leftarrow$ ParetoBlock($\{\rho_m(k) \forall m \in \mathscr{M}\}$), for the considered $k \in \mathscr{C}$ ▷ Revised Pareto-efficient fair secondary resource allocation adhered to budget constraint and allocated CPU
    $S'' = \{c_k, \rho'_k(m) \forall m \in \mathscr{M}\}, \forall k \in \mathscr{C}$
    $S^\star = S''$ ▷ Output: Fair Pareto-efficient solution
else if $\sum_{k \in \mathscr{C}} c_k > B_c$ **then** ▷ Primary resource budget constraint not met
    $\{c'_k\} \leftarrow$ ParetoBlock($\{c_k\}$), $\forall k \in \mathscr{C}$ ▷ Revised Pareto-efficient fair primary resource allocation adhered to its budget constraint
    $\{\rho'_m(k) \forall m \in \mathscr{M}\} \leftarrow$ ParetoBlock($\{\rho_m(k) \forall m \in \mathscr{M}\}$), $\forall k \in \mathscr{C}$ ▷ Revised Pareto-efficient fair secondary resource allocation adhering to revised Pareto efficient fair primary resource allocation
    $S'' = \{c'_k, \rho'_m(k) \forall m \in \mathscr{M}\}, \forall k \in \mathscr{C} S^\star = S''$ ▷ Output: Fair Pareto-efficient solution

Algorithm 3 ParetoBlock

Input: : $\{c_k\} \forall k \in \mathscr{C}$ s.t. $\sum_{k \in \mathscr{C}} c_k > B_c$ or $\{\rho_m(k) \forall m \in \mathscr{M}\}$ s.t. $\sum_{m \in \mathscr{M}} \rho_m(k) > B_\rho$ ▷ Primary or secondary resource allocation violating the budget constraints
$S_1 = \{c_k\} \forall k \in \mathscr{C}$ or $S_1 = \{\rho_m(k)\} \forall m \in \mathscr{M}$, as applicable $\mathscr{S}_e = \{S_1, S_2, \ldots\}$ ▷ Build expanded solution set
$\mathscr{S}_s \leftarrow \{S_i / |\mathscr{C}|\}_{\mathscr{S}_e}$ or $\mathscr{S}_s \leftarrow \{S_i / |\mathscr{M}|\}_{\mathscr{S}_e}$, as applicable ▷ Rescale expanded solution set
for $S \in \mathscr{S}_s$ **do** $\widehat{\mathscr{S}_s} \leftarrow \{\widehat{a}_k(S)\}$ ▷ Define refined actions set wrt $\mathscr{S}_s$
Create $\mathscr{S}_d$ ▷ Pareto dominant solution set
Choose S'_1 ▷ Fair Pareto-efficient resource allocation
Return: $S'_1 = \{c'_k\} \forall k \in \mathscr{C}$ or $S'_1 = \{\rho'_m(k)\} \forall m \in \mathscr{M}$, as applicable =0

It is important to note here that not only the CPU allocation across the services and RB allocation across the UEs are capacity constrained, the RB allocation is also dependent on the CPU allocated to vRAN. Thus, CPU and RB (resp.) act as the primary and secondary capacity constrained resources for vRAN. Unlike vRAN, the livecast service has no associated secondary capacity constrained resource. However, for the sake of mathematical proofs, this observation can be generalized as follows: the primary capacity constrained resource (here CPU) is distributed among K services, and each service may in turn serve M units (UEs for vRAN, none for livecast) using the primary resource. The secondary capacity constrained resource is distributed among M units of the service (if applicable). Further, the QoS satisfaction of each service in entirety comprises QoS satisfaction of individual units, and depends both on primary and secondary capacity constrained resource allocation.

To model such interdependence, we formulate the fair Pareto-efficient allocation of CPU across the services and RBs across the UEs in a given decision window as a constrained joint multi-criteria optimization problem. Further, for notational simplicity, we assume that each decision window comprises of just one monitoring

slot, i.e., $N = 1$. Let $\mathscr{C}, \mathscr{M}$ denote the set of services and UEs; given a set of coefficients $u_k \geq 0, k \in \mathscr{C}, m \in \mathscr{M}$, with $\sum_{k\in\mathscr{C}} u_k = 1$, it is required to find a solution $S^\star = \{c_k^\star, \rho_m^\star(k) \ \ \forall m \in \mathscr{M}\}, \forall k \in \mathscr{C}$, that maximizes $\sum_{k\in\mathscr{C}} u_k \Gamma_k(S)$ such that $S \in \mathscr{S}_c$, $\sum_{k\in\mathscr{C}} c_k \leq B_c$ and $\sum_{m\in\mathscr{M}} \rho_m(k) \leq B_\rho$. Here, c_k is the primary capacity constrained resource allocated to k-th service, $\rho_m(k)$ denotes the secondary capacity constrained resource allocated to m-th unit of the k-th service, $\mathscr{S}_c$ is the set of feasible capacity constrained resource allocations and $\Gamma_k(S)$ is the criteria function denoting the reward of the k-th service in a decision period following the CPU allocation strategy S.

The flow of fair Pareto-efficient resource allocation is summarized in Alg. 2. The key component of Alg. 2 is ParetoBlock (Alg. 3) that solves the joint multi-criteria optimization problem. It is invoked whenever the sum of allocated primary (secondary) capacity constrained resource exceeds its specified budget. It initially considers the CPU (RB) allocation to the services (UEs) provided by the greedy resource allocation policy, and creates the expanded CPU (RB) allocation solution set by considering all possible values for the c_k's ($\rho_m(k)$'s) that are greater than those output by the greedy policy and whose sum does not exceed $|\mathscr{C}|$ ($|\mathscr{M}|$) times the available budget. Such values are then scaled by $|\mathscr{C}|$ ($|\mathscr{M}|$), to get candidate allocation values that meet the CPU (RB) budget. The corresponding action set, $\widehat{\mathscr{S}_s}$, is built starting from such c_k's ($\rho_m(k)$'s) and possibly refining the actions so that their components take feasible values considering the dependence of primary and secondary resource. Such actions, $\{\widehat{a}_k(S)\}$, $\mathscr{S} \in \widehat{\mathscr{S}_s}$, are then used to compute the values of $\Gamma_k(S)$ to identify the Pareto-dominant solution set through iterative search and update, $\mathscr{S}_d \leftarrow \{S\}$, s.t. $\forall S \in \mathscr{S}_d, \forall S' \in \widehat{\mathscr{S}_s}, \Gamma_i(S) > \Gamma_i(S'), \Gamma_j(S) \geq \Gamma_j(S'), \forall i, j \in \mathscr{C}(\mathscr{M}), i \neq j$. Finally, for the primary capacity constrained resource, the Pareto-dominant solution that maximizes the minimum value of criterion function, i.e., the reward value over all the services is chosen as the fair Pareto-efficient solution. For the secondary capacity constrained resource, we define $g(m) := l_t(m) - l_i(m) \ \ \forall m \in \mathscr{M}$, where $l_t(m), l_i(m)$ denote (resp.) the target traffic load and the instantaneous rate achieved by m-th UE. Further, we choose the secondary resource allocation $\{\rho'_m(k) \ \ \forall m \in \mathscr{M}\}$ for a given fair primary resource c'_k as the solution that minimizes $\max_{m\in\mathscr{M}} g(m)$ over all $S \in \mathscr{S}_d$.

We finally prove the following results:

- The solution $S^\star$ introduced above is Pareto-efficient with respect to the primary (see Proposition 1), as well as jointly Pareto-efficient with respect to primary as well as secondary capacity constrained resources (see Proposition 2);
- Alg. 3 converges to a Pareto-efficient solution set, at a sub-linear rate (see Proposition 3);
- Alg. 3 converges to a solution that is fair with respect to the primary capacity constrained resource (see Proposition 4), as well as the secondary capacity constrained resource for a given primary capacity constrained resource allocation (see Proposition 5), thus leading to a fair Pareto-efficient solution.

Proposition 1. *Pareto-efficient allocation of the primary resource*: Given a set of coefficients $u_k \geq 0, k \in \mathscr{C}$, s.t., $\sum_{k\in\mathscr{C}} u_k = 1$, the solution $S^\star = \{c_k^\star\}, k \in \mathscr{C}$, maximizing the multi-criteria optimization problem $\sum_{k\in\mathscr{C}} u_k \Gamma_k(S)$, is Pareto-efficient.

Proof. See Appendix C [1]. □

Proposition 2. *Pareto-efficient joint allocation of primary and secondary resource*: Given a set of coefficients $u_k \geq 0, k \in \mathscr{C}$, such that, $\sum_{k\in\mathscr{C}} u_k = 1$, then the solution $S^\star = \{c_k^\star, \rho_m^\star(k) \ \ \forall m \in \mathscr{M}\}, \forall k \in \mathscr{C}$, that maximizes the multi-criteria optimization problem $\sum_{k\in\mathscr{C}} u_k \Gamma_k(S)$, is Pareto-efficient.

Proof. See Appendix C [1] □

Proposition 3. Alg. 3 converges to a Pareto-efficient solution set at a sub-linear rate.

Proof. See Appendix C [1] □

Proposition 4. *Fairness of Pareto-efficient primary resource allocation*: The solution $S^\star = \{c_k^\star, \rho_m^\star(k) \ \ \forall m \in \mathscr{M}\}, \forall k \in \mathscr{C}$ obtained using Algorithm 2 is fair with respect to primary resource allocation $c_k^\star, \forall k \in \mathscr{C}$.

Proof. See Appendix C [1] □

Proposition 5. *Fairness of Pareto-efficient secondary resource allocation in the vRAN*: For a given fair primary resource allocation $c_k^\star$ in the solution $S^\star = \{c_k^\star, \rho_m^\star(k) \ \ \forall m \in \mathscr{M}\}$, for $k = 2$ (denoting the vRAN service), $S^\star$ is fair with respect to secondary resource allocation $\{\rho_k^\star(m) \ \ \forall m \in \mathscr{M}\}$.

[1] https://github.com/somreetakgp/VERA-Supplementary-Material

Proof. See Appendix C [1] □

We remark that two-stage solutions for resource allocation via distributed RL have often been applied in the literature, where greedy global solutions are obtained in the first stage, and then each agent tunes its own parameters in the second stage to ensure fairness. However, compared to such existing schemes, our proposed VERA framework has several unique features: (i) it jointly considers capacity-constrained resources and service specific operating parameters, (ii) it accounts for the interdependence of primary and secondary capacity-constrained resources, (iii) it envisions a novel Pareto block design for ensuring Pareto-optimal fair action selection for virtual services hosted at the network edge. To the best of our knowledge, none of the existing works have addressed these issues.

4.4.4 Learning algorithm

We exploit the concept of experience-based learning using sample sequences of shared context, actions, and rewards observed from the actual interaction of the RL agent with the environment. SARSA, an acronym for quintuple $(S_t, A_t, R_t, S_{t+1}, A_{t+1})$, is an on-line policy algorithm where learning of the RL agent at time t is governed by its current state S_t, choice of action A_t, reward R_t received on taking action A_t, state S_{t+1} that the RL agent enters after taking action A_t, and finally the next action A_{t+1} that the agent chooses in new state S_{t+1} [75]. We here highlight that compared to Q-learning, another commonly used learning algorithm, SARSA adopts a conservative learning approach by avoiding high-risk actions that may generate large negative rewards from the environment. This is especially relevant since the applicability of VERA framework may be extended to ultra low latency and ultra reliable services wherein ensuring radio connectivity is critical and any violation of KPI targets would incur high costs. Although double Q-learning, a Q-learning variant, is also conservative, it is computationally more intensive than SARSA owing to the requirement of computing and updating two Q-policies simultaneously. Further, SARSA has low per-sample variance, which makes it less susceptible to convergence problems [75]; hence, future extensions of the VERA framework can be easily upgraded to deep networks if necessary.

For clarity and without loss of generality, we focus on the learning of a single RL agent that corresponds to one of the services, over successive decision windows.

Given the mean shared context and possible actions, the primary steps in the learning algorithm are: (i) obtain greedy resource allocation policy for service k through estimation of action values $q_{\pi_k}(y,a)$, (ii) obtain a fair Pareto-efficient resource allocation policy by collating and returning the greedy policies of all the services, and (iii) update of the action-value estimates for service k using differential semi-gradient SARSA [75].

4.4.5 Computational complexity analysis

We now discuss the complexity analysis of the VERA framework implementation. The most complex operations in the whole framework are given by the following steps: (i) greedy action selection for the K services, (ii) joint Pareto-efficient fair allocation of primary resource among K services, and secondary resources among M units of each service, (iii) computation of weighted mean of the context and mean reward for K services in a decision window comprising N monitoring slots, and (iv) update of the weight vector for learning radio policy based on the KPIs observation from the K services. Corresponding to each of these steps, the computational complexities of taking the resource allocation decision once in the VERA framework are given by $\mathcal{O}(|\mathcal{A}|)$ (with $|\mathcal{A}|$ being the cardinality of the VERA action space), $\mathcal{O}(KM)$, $\mathcal{O}(KNM)$, and $\mathcal{O}(K)$, respectively. Hence, the overall complexity is $\mathcal{O}(|\mathcal{A}|)+\mathcal{O}(KM)+\mathcal{O}(KNM)+\mathcal{O}(K) \approx \mathcal{O}(|\mathcal{A}|)$ since the first term is the dominant one as $|\mathcal{A}|$ is much larger than KNM. Thus, computations in VERA scale linearly with the increase in the cardinality of the action space.

4.5 Proof-of-concept Implementation

In this section, we first introduce our proof-of-concept implementation (Sec. 4.5.1), and then we present the parameters and settings we use to collect our datasets and run our experimental tests (Sec. 4.5.2).

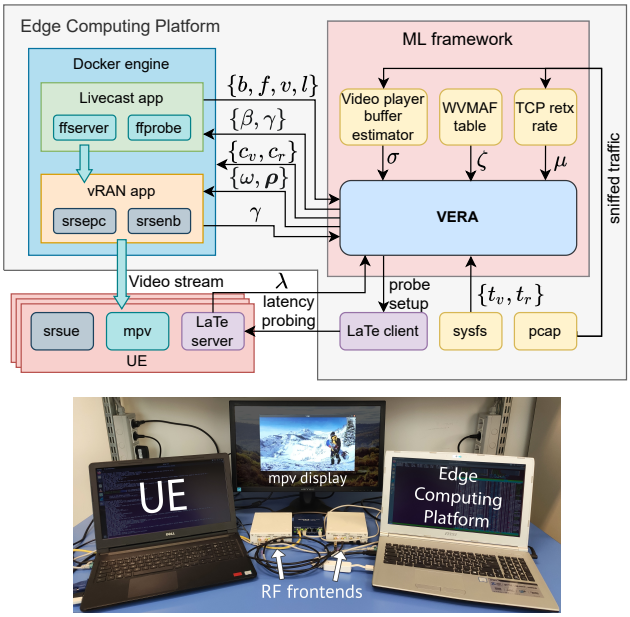


Fig. 4.9 (top) VERA system representation (details in Sec. 4.5.1); (bottom) picture of the testbed (details in Sec. 4.5.2)

4.5.1 VERA implementation

As depicted in Fig. 4.9, we have integrated VERA in a testbed based on `srsRAN` (to emulate a vRAN service), `ffserver` (to emulate a livecast video service), and `mpv` (livecast video client deployed at each UE video player).

VERA's learning agents receive real-time context information directly from the vRAN, the livecast service, and the edge computing platform, using custom-made TCP-based interfaces. More specifically, the CQI information is retrieved from srseNB using its `enb_metrics_interface` class. The input video features (i.e., input video FPS, bitrate, and resolution) are instead probed using `ffprobe`, which is part of `ffmpeg`, and then sent to VERA whenever a new video source is selected. The CPU throttling times of the livecast and vRAN services are collected by interfacing with Linux `cgroups`, using Linux' pseudo file-system `sysfs`. Finally, the network load, which corresponds to the offered load that the vBS has to process, is derived by summing up the bytes produced each second by the video encoder, as reported in its output log.

Concerning the reward signal, the observed unidirectional latency, the packet loss rate, the estimated video player buffer state, and the WVMAF are sent to

VERA in real time, using additional custom TCP-based interfaces. In more detail, the latency measurements are obtained using LaTe[4], a flexible client-server multi-protocol Latency Tester that sends probes to the network under test and measures the delay that the probe experiences. To this end, clock synchronization between the Edge Platform and the UEs is performed using Precision Time Protocol daemon (PTPd).

The packet loss rate, computed through the TCP segment retransmission rate, and the buffer occupancy are estimated at the edge platform by leveraging, respectively, the TCP sequence and the acknowledgment numbers obtained using `libpcap`. To model the player buffer and, hence, infer its status, VERA exploits the TCP acknowledgment numbers, the offered load, and the output frame numbers of the video encoder, as explained in Sec. 4.4. Finally, to compute the WVMAF, we use a lookup table that maps every choice of encoding parameters to the expected VMAF score. Calculating the VMAF score is indeed a computing-intensive task that cannot be performed in real-time without a noticeable performance impact. The table has been built by considering a collection of 1080p video samples[5], and by encoding each source video using every encoding parameters combination available to VERA. The VMAF score is calculated by comparing the frames of each encoded video to the original frames. The WVMAF score of each video sample is obtained by multiplying its VMAF score by the ratio of the output frame rate and the input frame rate. Then, for every combination of the encoding parameters, the corresponding WVMAF score is computed averaging the WVMAF scores of all video samples encoded with the same combination of parameters.

At last, to enforce decisions made by VERA, the per-UE RB allocation and the MCS policy are sent to the vRAN through a custom interface, the CPU allocation is set through Docker API (which, in turn, enforces it using Linux cgroups), and the video encoding FPS and bit rate policies are updated overriding the parameter settings in the livecast service, thus allowing VERA to work in real-time.

[4] https://github.com/francescoraves483/LaMP_LaTe
[5] https://media.xiph.org/video/derf/

Table 4.2 Input and output video characteristics

Video characteristics	**Input**	**Output**
Resolution [pixels]	1920×1080	1280×720
Bit rate [Mbps]	18	0.3, 1.5, 5, 10
Frame rate [FPS]	30	10, 20, 30
Codec (container)	VP9 – WebM	VP9 – WebM

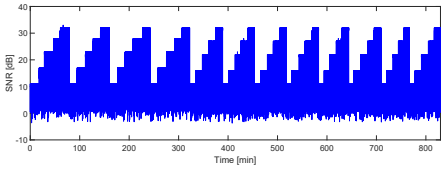


Fig. 4.10 Time evolution of SNR in our experiments

4.5.2 Testbed configuration

The testbed configuration used to collect the necessary dataset and run the VERA framework in real time is based on the architecture presented in Sec. 4.2.

The edge computing platform and the UEs are hosted on GNU/Linux machines; they accommodate, respectively, an Intel i7-7700HQ and an Intel i7-8550U CPU, with 16 GB of DDR4 memory. The LTE network uses a 10-MHz channel in band 7, which provides a capacity of 50 RBs. The dynamic SNR pattern, considered in our experiments to be experienced by the UEs, is depicted in Fig. 4.10; the values of SNR are then mapped into CQI by the vRAN system. We assume a network slice dedicated to the livecast service, with a capacity that may vary between 12 and 36 RBs. As RF frontend, Ettus USRP B210 boards are used to perform up/down-conversion, filtering, amplification, and AD/DA conversion of the UEs and eNB LTE signals.

As mentioned above, we use `ffmpeg`, as this is compatible with a wide set of video codecs, picture formats, containers, besides offering a number of filters to modify video characteristics. Tab. 4.2 reports the characteristics of the input and output videos in our experiments. The characteristics of the input videos are representative of a livecast content, as they ensure that the video can be properly played by the client player in a wide range of network conditions and client configurations. The output parameters have been set so as to allow for the best video quality retention, hence a good level of user Quality of Experience, while requiring a reasonable consumption of network and computing resources. Tab. 4.3 includes additional parameters settings

Table 4.3 Video encoder, server and client parameters

Encoder/server param.	**Description**	**Value**
StartSendOnKey	If set, the video is streamed starting from the first I-frame generated by the encoder, i.e., P-frames not preceded by an I-frame are discarded	Enabled
Preroll N	The video is streamed starting not from the most recent frame but from N seconds in the past; if set, it increases buffer occupancy at the expense of the end-to-end latency	Disabled
VP9 Threads	No. of threads that decoder & encoder can use: high values increase speed if multiple vCPUs are allocated, at the cost of a small overhead.	No. of allocated vCPUs
VP9 Quality	Possible settings: realtime, good, or best. It controls the time that the encoder can take to encode frames beyond their presentation time	Realtime (no additional time beyond presentation timestamp)
VP9 Speed	It controls the trade-off between computational lightness and picture quality. Possible values in [0,16], with the higher values prioritizing encoding speed (i.e., lower CPU consumption) over picture quality	16
Client CachePauseInitial	The client pauses the playback at the beginning to wait for the buffer to fill, so as to avoid pauses while the video is playing	Enabled
Client CachePauseWait	Video time that the client requires before resuming the playback when paused. It affects the end-to-end latency, but a bigger size can better cope with oscillations in the data transfer and encoding delays	1 s

that we have used at the video encoder, server, and client, which are typical of a livecast service.

Finally, we consider that decisions are made every monitoring slot ($N = 1$), and, unless otherwise specified, we set the available CPU budget to 3 vCPUs. We would like to highlight that the computation cost of our solution can be fully sustained by our small-size testbed. On average, one iteration (i.e., metrics parsing, action selection, reward computation and weights update) requires 16.1 ms. Over one thousand iterations, we measured a maximum iteration time of 25.8 ms, with a 99-th percentile below 18.6 ms. This corresponds to an average CPU load below 5% when the monitoring slot is 100-ms long.

4.6 Evaluation and experimental validation

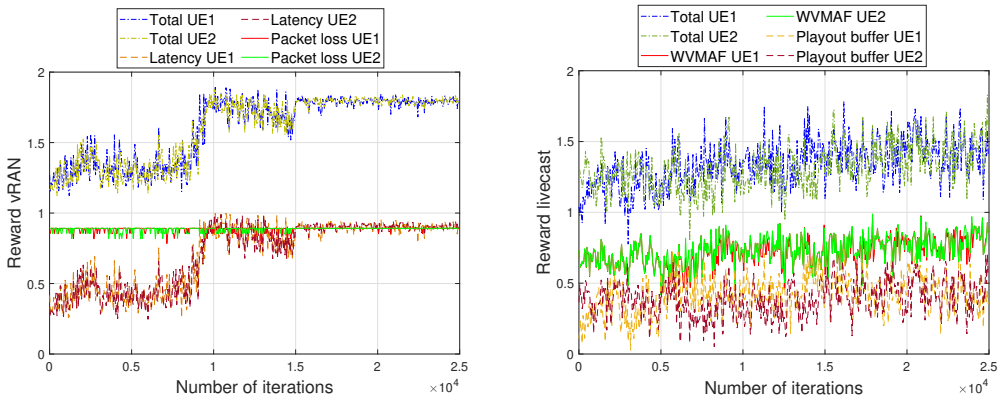
In this section, we first present the numerical results (Sec. 4.6.1) derived using the data sets obtained through extensive experiments on the testbed described in Sec. 4.5; and then, we present the performance of a real-time implementation of VERA on the testbed (Sec. 4.6.2).

4.6.1 Numerical results

The baseline scenario we consider in our numerical performance evaluation includes 1 vBS, 2 UEs, and a livecast service streaming a single video to both UEs. The CPU and RB budgets are fixed to 2 vCPUs and 60 RBs, respectively.

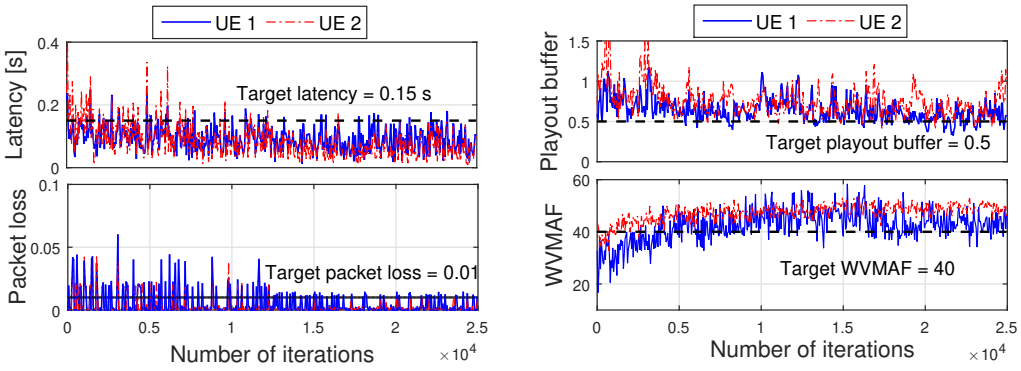
Convergence evaluation. Fig. 4.11 depicts the time evolution of reward values for the vRAN and livecast services in the baseline scenario. From the plots, we observe that, despite the large heterogeneous action set and the diverse context vector, the reward corresponding to each of the KPIs, and hence the total reward, saturates close to the maximum value for both the UEs, thereby highlighting the efficient learning capability of the VERA framework. Also, the convergence of the livecast service is relatively slower with respect to vRAN owing to its slowly varying dynamics.

KPI performance. Next, Fig. 4.12 presents the evolution of the KPIs across iterations during the learning process. Notice that the KPI satisfaction for vRAN is



(a) (b)

Fig. 4.11 Convergence of reward values: vRAN (a) and livecast (b) services



(a) (b)

Fig. 4.12 KPI evolution with respect to iterations: (a) vRAN and (b) livecast. Dashed dark line shows target KPI values.

achieved when its latency and packet loss do not exceed their respective targets. On the contrary, for the livecast service, the playout buffer and WVMAF should not fall below their target values, while keeping the KPIs observed for both the services as close as possible to their target values. According to the 3GPP 5G specifications [29] and acceptable QoE, the target KPI values are set at 150 ms, 0.01, 0.5 s and 40 (resp.) for latency, packet loss, playout buffer, and WVMAF. From the plots, we observe that barring a few initial iterations during which the algorithm is still learning, the choice of actions by the VERA framework leads to KPI satisfaction for both vRAN and livecast services. To quantify VERA's suboptimality, the mean KPI target violation for VERA post convergence of the algorithm is 3.7%.

Performance under different constraints. We now evaluate the impact that different CPU and RB capacity constraints have on the performance of VERA.

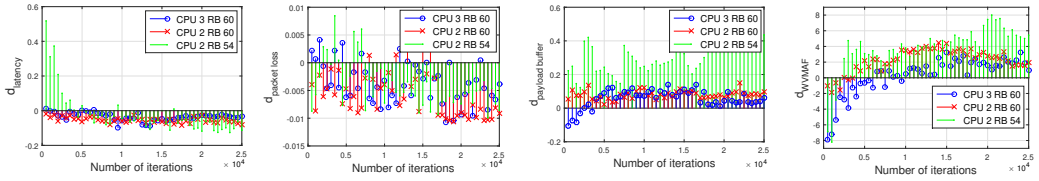


Fig. 4.13 Performance of VERA under varying CPU and RB budget constraints: (a) latency, (b) packet loss, (c) playout buffer, and (d) WVMAF

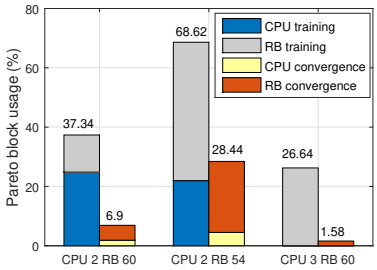


Fig. 4.14 Utilization of Pareto block in VERA during training and after convergence has been attained

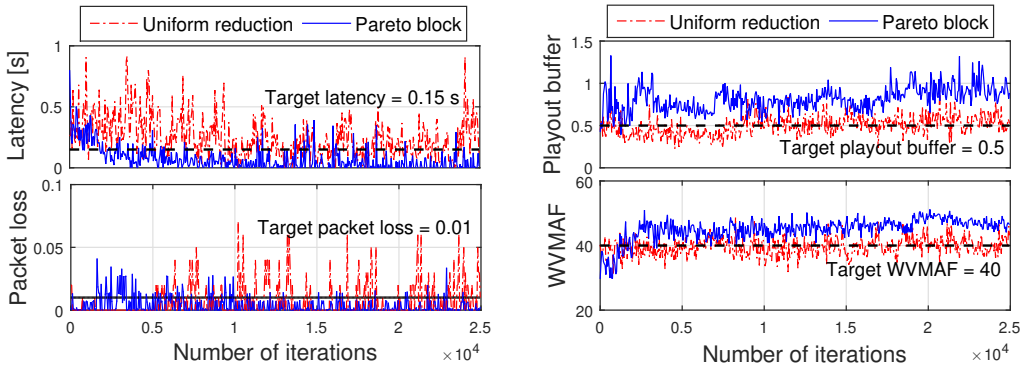


Fig. 4.15 KPI comparison of VERA with Pareto block and uniform reduction approach: (a) vRAN, (b) livecast

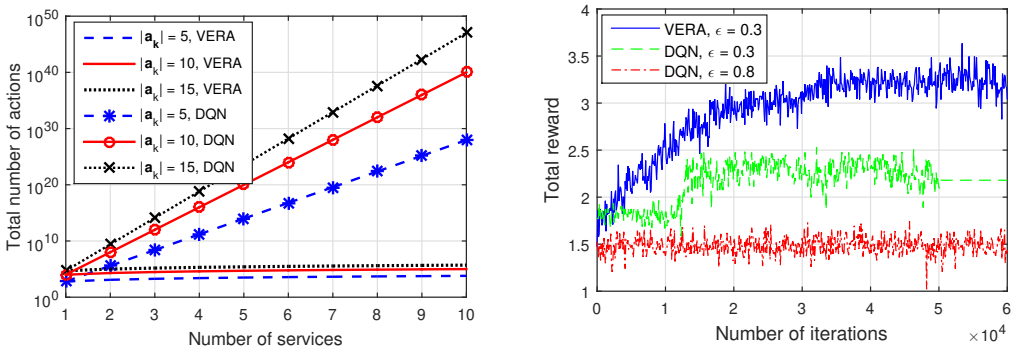
To this end, we consider two additional scenarios having budgets 3 vCPUs - 60 RBs and 2 vCPUs - 54 RBs, along with the baseline scenario. The performance is characterized using a distance parameter $d_{KPI} = \text{KPI}_o - \text{KPI}_t$, where KPI_o is the KPI target threshold and KPI_t is the KPI experienced at time t, and which basically quantifies how far away the observed KPI value is from its target. Fig. 4.13 compares the average value of d_{KPI} computed over both UEs for the said scenarios, for both vRAN and livecast services. It may be noted that a negative (positive) value of d_{KPI} for vRAN (livecast) denotes KPI satisfaction, and, irrespective of the service type, it

is desirable that d_{KPI} is as close as possible to 0. From the plots, we observe that when the budget constraints are stringent, the choice of actions in order to meet the KPI target values is limited. Consequently, the resource allocation efficiency is slightly compromised as shown by higher d_{KPI} values. Nevertheless, the KPI satisfaction is still achieved. As more resources are made available in terms of CPU and RBs, d_{KPI} values cling closer to 0, thereby minimizing the wastage in resource allocation. Thus, VERA can successfully attain KPI satisfaction for both the services and UEs under varying CPU and RB capacity constraints, however, stringent constraints may lead to a marginal loss in efficiency.

Pareto block statistics. Next, we investigate the significance of the Pareto block in the VERA framework. The bar plot in Fig. 4.14 shows the statistics of the Pareto block usage under different CPU and RB capacity constraints. We observe that the Pareto block usage for CPU as well as RB is the highest when budget is the most stringent, i.e., 2 vCPUs - 54 RBs. However, on a positive note, the Pareto block invocation substantially reduces once VERA has attained convergence compared to its training phase. This in turn suggests that, when a pre-trained VERA model is used, the Pareto block will not add to the system runtime complexity.

To further emphasize this aspect, we replace the Pareto analysis in the VERA framework by a more intuitive and simpler uniform reduction approach, wherein an equal proportion of any excess CPU (RB) allocated beyond the budget is subtracted from the allocated CPU (RB) values across the services (UEs) such that the budget constraint is met. Fig. 4.15 presents the KPI (averaged over both UEs) comparison using the Pareto block and uniform reduction in the worst of our considered scenarios, i.e., 2 vCPUs - 54 RBs for vRAN and livecast services. From the plots we observe that unlike the Pareto block, uniform reduction fails to meet the target KPI values. This confirms that the Pareto block has a crucial role in optimal resource orchestration, especially when resources are constrained.

Comparison with other approaches. Finally, we address the scalability of the VERA framework. To emphasize the distributed decision making used by VERA, we compare its performance to a data-driven centralized framework using deep Q-network (DQN). Since a generic DQN has no provision to enforce capacity constraints, to be fair, we consider a scenario wherein full CPU is available to the hosted services and there is no RB capacity constraint.



(a) (b)

Fig. 4.16 Performance of VERA in comparison to DQN: (a) cardinality of action space, (b) total reward

Before discussing the numerical results, we first characterize the scalability of VERA and DQN as follows. We generalize the notation of action vector for k-th service as $a_k = \{a_{k1}, a_{k2}, \cdots, a_{kP}\}$, with the generic a_{kp} indicating the p-th action variable of the k-th service, which can assume values from a set of $N_{a_{kp}}$ discrete elements, with $P = |a_k|$. Since DQN is a centralized framework, it handles the action vector corresponding to different services together, consequently, $|\mathscr{A}|_{DQN} = \prod_{k=1}^{K} \prod_{p=1}^{|a_k|} N_{a_{kp}}$. On the contrary, VERA handles different services in a distributed manner by assigning a separate learning agent for each service, leading to $|\mathscr{A}|_{VERA} = \sum_{k=1}^{K} \prod_{p=1}^{|a_k|} N_{a_{kp}}$. For simplicity, assuming that all the services comprise the same number of action variables $|a_k|$, and each action variable assumes a value from the set of discrete elements of the same cardinality $N_{a_{kp}}$, then $|\mathscr{A}|_{DQN} = N_{a_{kp}}^{|a_k|^K}$, and $|\mathscr{A}|_{VERA} = K \times N_{a_{kp}}^{|a_k|}$. Fig. 4.16a shows the variation of the total number of actions in VERA and DQN frameworks with increasing number of services, for $N_{a_{kp}} = 4$. It can be clearly observed that as the number of services hosted in the server increases, the cardinality of the action space for DQN rises very sharply in comparison to that of VERA, and this difference is even more evident as the number of action variables per service also increases.

Further, Fig. 4.16b shows the convergence of total reward from vRAN as well as livecast services observed in consequence to actions chosen by VERA and DQN. The choice of reward as a comparison metric helps to better characterize the scalability of the system. We observe that due to the high cardinality of the action space in DQN, it converges poorly. With the same ε-greedy action selection policy as VERA

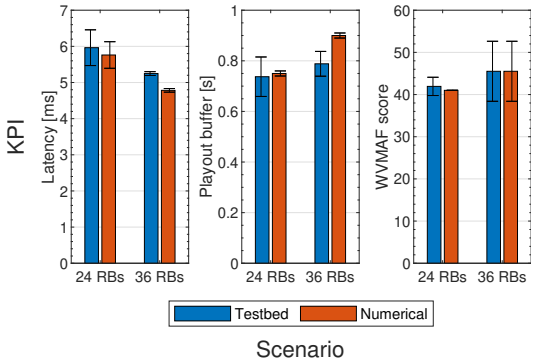


Fig. 4.17 Comparison of testbed and numerical KPI values for the livecast and vRAN services on the 24 RBs and 36 RBs scenarios with CPU budget equal to 3 vCPUs. Error bars indicate the confidence interval at 95% confidence level.

(i.e., $\varepsilon = 0.3$, ε-decay $= 0.9999$), DQN is unable to explore all the actions while ε decays down to a negligibly small value and the learning agent gets caught up in a local optimum. Even if the action selection parameters are improved ($\varepsilon = 0.8$, ε-decay $= 0.9999$), DQN still explores the action space until $\sim$ 60k iterations in our experiments, whereas VERA has shown a clean and early convergence with smaller ε value. To this end, it is worth noting that VERA exhibits a much higher scalability in terms of cardinality of action space compared to the state-of-the-art DQN-based centralized framework. To quantify the scalability performance, we define the scaling cost as the sum of reward deficit with respect to maximum reward value at convergence and the fraction of total iterations used for convergence. In our experiments, we found that the scaling cost of VERA is 45% and 60% lower compared to DQN, (resp.) for $\varepsilon = 0.3$ and $\varepsilon = 0.8$. We also compare the KPI performance of VERA with respect to DQN, which is included in the supplemental material due to space limitations here.

4.6.2 Proof-of-concept results

A pre-trained version of the VERA RL agents has been generated offline using the dataset collected from our testbed, then it has been evaluated online and in real time in two scenarios with different RBs availability, i.e., 24 and 36 RBs, and with CPU budget equal to 3 vCPUs. Notice that the use of pre-trained RL agents is only to expedite the learning curve at the inception of VERA framework in the testbed. In a real-world scenario, VERA uses differential semi-gradient SARSA for continuously

learning from its environment over the time horizon. In both scenarios, a single UE connects to the vRAN and receives the livecast, using the configuration described in Sec. 4.5.2. For this case, the USRPs' transmission gain is set to a high value, so as to ensure that the SNR on uplink and downlink does not drop below 29 dB.

The radio latency, playout buffer and WMAF KPIs, collected from both testbed and numerical experiments, are compared in Fig. 4.17 (packet loss is omitted as it is equal to 0 in all cases). All KPIs are always satisfied for both the vRAN and the livecast service. As expected, the latency is higher in the case of 24 RBs than for 36 RBs, on the contrary, the playout buffer and WVMAF are less for 24 RBs than 36 RBs, owing to the lower number of radio resources being available in the first scenario: a higher number of allocated RBs leads to higher user throughput, which results in less latency, higher playout buffer and higher WVMAF score.

Importantly, the relative difference between testbed and numerical KPI values never exceeds 12.4%, with such a value being observed in the case of the playout buffer in the 36 RBs scenario. The similarity between testbed and numerical results validates VERA performance in a real time, hardware-in-the-loop implementation, and it demonstrates the effectiveness of our solution in a real-world environment.

4.7 Related Work

Several works have addressed the VNF placement problem at the network edge, which is related but orthogonal to the problem we face. Recent examples include: [78], which minimizes latency and system cost; [79], which optimizes both service placement and traffic routing under different resource constraints; and [80], which uses cooperation among edge nodes for service caching and workload scheduling.

Other studies have focused on QoE provisioning to mobile users through edge-assisted solutions. In particular, [81] presents an RL framework for crowdcasting services at the edge meeting bit rate as well as streaming and channel switching latency requirements, while minimizing the overall computing and bandwidth cost. [82], instead, designs and implements an edge network orchestrator, and a server assignment and frame resolution selection algorithm for best latency-accuracy trade-off in mobile augmented reality.

Relevant to our work are also existing studies on resource consumption by edge user applications. In particular, [83] investigates the impact of real-time video analytics on computing and energy resources, while [84] focuses on image processing through CNNs and maximizes the learning accuracy given the limited resources at the edge.

The above literature does not consider the inherent resource contention between edge services and virtualized RANs. Related with this, [68] jointly optimizes the allocated resources to edge services and the placement of RAN functions, but uses an over-simplistic linear optimization model that cannot adapt to quick system dynamics. Like us, some other authors have used machine learning for practical resource allocation, radio parameter settings, and service KPI support in cellular networks. Among these, [85–89] aim at maximizing throughput through channel or link-rate selection, using multi-armed bandit techniques. Similarly, but leveraging the contextual information from the environment, [90] proposes an RL approach for rate selection and resource allocation, and [70] to maximize throughput subject to power consumption constraints. Finally, [69] extends the latter to accommodate service KPI constraints. However, [69] does not consider individual user dynamics nor fairness in resource allocation, as we do. RL-based schemes can also be found in [91–93], to minimize latency and packet drop rate in 5G systems. The work in [17], instead, tackles computing resource allocation in a virtualized radio access, and introduces a deep RL approach for resource management. Deep RL is also used to determine the suitable MCS and transmit power level in cognitive radio networks in [94] and [95], respectively, and to maximize the network sum-rate in [96]. More recently, [97] proposed a data-driven O-RAN-compliant framework that configures DUs/RUs according to a specified spectrum access policy.

We underline that, unlike previous work, we address the allocation of edge resources *constrained to a limited budget across different, competing, virtual services.* Through our testbed, we identify the non-trivial correlations existing among the actions related to the different services, making the VERA learning objectives very different from those of existing works. To derive a scalable solution that can accommodate multiple services and vBSs (or slices of vBSs), we resort to a multi-agent RL mode. And, inspired by [71] and other literature on autonomous driving, we accommodate such hard constraints as a non-learnable building block. Different from previous work, however, we design a Pareto-efficient block for this task, which provides fair resource allocation across agents (vBSs and edge applications).

Finally, a preliminary version of our solution with only one user and considering only CPU as a constrained resource was presented in our conference paper [98].

4.8 Conclusions

We considered an edge computing platform hosting virtualized user applications and network services (namely, vRAN) competing for the same resources. We first investigated the correlations existing between the dynamics of such services through an experimental testbed that leverages a containerized livecast application and a containerized LTE base station. Then we developed a distributed learning framework, called VERA, that sets the configuration of both types of services so that the target KPIs can be met in spite of the limited availability of computing resources at the edge. Importantly, VERA also exploits a Pareto analysis that leads to fair Pareto-efficient decisions, and it can scale well with the number of virtualized services that are hosted at the edge platform. Our experimental results demonstrate the feasibility of the VERA approach and the important role of the Pareto analysis. Also, they show the excellent performance we can obtain in the presence of capacity-constrained resources with the KPI target violation limited to just 3.7%. Further, we show that VERA performs similarly when executed in our real-time proof-of-concept implementation, with KPI differences below 12.4%, thus confirming the effectiveness of VERA also in a real-world environment. Finally, we compare the performance of VERA to the centralized DQN framework and found it to be 54% more scalable, thereby establishing the efficacy of distributed over centralized learning in such complex resource limited scenarios.

Chapter 5

Conclusions

In this thesis, resource orchestration solutions are designed and developed for virtualized 5G mobile networks. The thesis explored underlying concepts of Network Function Virtualization (NFV), Mobile Edge Computing (MEC), Network slicing, and Machine Learning. By encompassing them into the work, their potential to enhance efficiency, adaptability, and network intelligence is effectively demonstrated. This opens up new corridors for vertical industries to deliver cutting-edge services to the ever-evolving mobile users.

To respond to the research questions in Sec 1.1, resource orchestration solutions are designed with a focus on model-based and model-free approaches. Towards this goal, we first gain a hands-on understanding of vRAN behavior, exploring the relationship between radio and computing resource dynamics. Through experiments conducted using our developed vRAN testbed, resource consumption under different settings and a varying number of users were assessed. Our findings provide crucial insights into the vRAN behavior, showing a significant increase in CPU and memory utilization of the RAN with the growing number of users. Regression models have been developed to anticipate system behavior as the user count rises, encompassing various radio transmission configurations. This methodology and the resulting prediction models serve as valuable tools for the design and optimization of virtual RANs. Then, a model-based RAN slicing strategy is considered for service provisioning in 5G/B5G networks. A cost-efficient slicing solution is designed to account for the real-world dependency between the cost of computing resources at the network edge and the ability of the RAN to support different network slices. To

accomplish this task two different slices, namely eMBB and uRLLC slices have been considered. The numerical results confirmed that our solution leads to a cost-efficient resource slicing with a 10-15% reduction in radio resource consumption, while also accomplishing performance isolation and meeting the data rate and delay specified in the SLAs of, respectively, eMBB and uRLLC slices. Finally, we investigated managing network and compute resources at the edge using reinforcement learning, aiming to achieve predefined KPIs. VERA employs a reinforcement learning framework to dynamically allocate resources to network services and user applications. Considering an LTE vRAN as a network service and an adaptive video transcoder as a user application, we demonstrated the ML framework's ability to learn the relationship between radio and video actions. To ensure fairness amid competing resource requests, VERA integrates Pareto analysis to prevent resource hogging. Our observations indicate that VERA consistently meets target KPIs for over 96% of observation periods.

Comparatively better performance of the proposed CES strategy (discussed in Chapter 3) for both orthogonal and non-orthogonal puncturing strategies encourages extending the work to non-orthogonal multiple access networks (NOMA), a key candidate technology for fifth-generation (5G) networks [99, 100]. The fundamental concept of NOMA involves serving multiple users in the same resource block (RB) in terms of time, frequency, or code. With signals of different users superimposed in the power domain, receivers leverage successive interference cancellation (SIC) to distinguish each other. Consequently, both the number of users and spectrum efficiency can be significantly improved [101].

With the evolution of mobile standards towards 6G, another frontier of innovation emerges in the form of the open radio access network (O-RAN). By disaggregating RAN components and opening up interfaces, O-RAN is regarded as the promising approach to transform wireless technology from "connected things" to "connected intelligence." The O-RAN architecture, enabling radio parameter tuning based on feedback from mobile terminals, aligns well with the proposed VERA framework (discussed in Chapter 4). The O-RAN alliance designed an architecture allowing non-real-time centralized intelligence through specialized applications called rApps. These can operate directly on the RAN or coordinate with distributed near-real-time applications, known as xApps, for more precise operations. The O-RAN architecture not only envisions the division between centralized non-real-time and distributed near-real-time decisions but also incorporates Machine Learning as a core component

in its specifications. This establishes the crucial role of this technology in shaping the future of mobile networks.

5.0.1 Future Work and Open Challenges

Sixth-generation (6G) systems are poised to usher in a new era characterized by the widespread adoption of massive and highly heterogeneous network slicing. This paradigm shift extends the tenancy model to end consumers, giving rise to a plethora of advanced and diverse digital services. This transformative landscape presents formidable challenges in network management and orchestration, with a focus on ensuring scalability and sustainability. Another latest frontier in this endeavour is O-RAN. The growing interest in the O-RAN architecture stems from its advantages in managing and optimizing the RAN. Its disaggregated, virtualized, and software-based components offer increased flexibility, reconfigurability, and resiliency in deploying the network infrastructure. Additionally, it allows network operators to use interoperable equipment from different vendors, reducing CAPEX and avoiding lock-ins. While addressing current operator needs, this paradigm introduces several research challenges, prompting further investigation.

Energy efficient massive network slicing- Enabling energy-efficient massive network slicing in future 6G networks necessitates architectural and algorithmic innovations [102]. The 6G RAN is poised to enhance energy efficiency (EE) through slice-level context awareness, employing AI-based analysis of slice KPIs, especially traffic patterns. These analytics may identify slices with complementary behaviors, like business and residential slices, optimizing the placement of their central units (CUs) as virtual network functions (VNFs) on the same server. This minimizes active servers and reduces energy costs. Moreover, the one-size-fits-all 5G functional split architecture is inefficient for a large number of RAN sub-slices. Adopting AI-driven dynamic distribution of RAN functions per slice improves resource utilization, performance, and RAN EE while meeting SLAs.

O-RAN architecture [103] involves diverse layers, controllers, VNFs, and splits requiring dynamic reconfiguration. The intricacy of this environment highlights AI-driven management using multi-agent learning as its key enabler. To support massive slicing, O-RAN architecture needs redesign to facilitate decentralized AI.

This poses challenges as heterogeneous agents may have different (and sometimes conflicting) objectives, actions and capabilities."

Slice-enabling cell-free- Cell-free massive MIMO [104] is a promising 6G wireless access technology known for high throughput and energy efficiency. It involves numerous distributed low-power single-antenna access points (APs) connected to a central network controller, posing a challenge for decentralized AI-based slice-level resource management tailored to each AP cluster."

Coexistence of virtualized services and vRAN-The VERA framework introduced in Chapter 4 is designed for optimizing the coexistence of video-streaming user services and vRAN at the edge. We demonstrated the intricate interdependence of resource demands between these two entities. This behavior, although potentially applicable to other user services, requires further research to explore the coexistence dynamics of various user applications and network services in the edge ecosystem. This exploration aims to identify potential opportunities for improved optimization.

Integration across multiple timescales- Network management applications operate at different timescales based on the specific network component they address. While the literature abounds with examples of rApps and xApps, the concept of RAN services constituted by multiple applications running at distinct time scales remains relatively unexplored. Investigating how these applications can synergize could lead to a deeper level of optimization and improved performance. O-RAN specifications provide clear examples of integrating different time scales within the lifecycle of Machine Learning models [103]. Notably, scenarios such as non-real-time ML model training, followed by its deployment and execution close to the user for near-real-time inference, present intriguing possibilities.

Evolving ORAN use cases- In Chapter 4 we explored video transcoding and streaming at the edge, alongside a vRAN, forming a learning framework for optimizing performance and resource usage. The O-RAN intelligent, data-driven closed-loop control architecture, aligned with O-RAN specifications, is well-matched for this framework. O-RAN specifications already cover this use case, and ongoing efforts are identifying and exploring new ones [105]. As these new use cases emerge, we will assess their compatibility with current O-RAN specifications and propose necessary additional features to ensure the O-RAN architecture supports its effective implementation.

References

[1] Ericsson mobility report. Technical report, June 2022.

[2] Insoo Hwang, Bongyong Song, and Samir S. Soliman. A holistic view on hyper-dense heterogeneous and small cell networks. *IEEE Communications Magazine*, 51(6):20–27, 2013.

[3] David Gesbert, Marios Kountouris, Robert W. Heath, Chan-byoung Chae, and Thomas Salzer. Shifting the mimo paradigm. *IEEE Signal Processing Magazine*, 24(5):36–46, 2007.

[4] Jakob Hoydis, Stephan ten Brink, and Mérouane Debbah. Massive mimo: How many antennas do we need? In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 545–550, 2011.

[5] Kulin Chen and Run Duan. C-ran the road towards green ran. *China Mobile Research Institute, white paper*, 2, 2011.

[6] Marketing Charts. Mobile network operators face cost crunch, 2015.

[7] Peter Rost, Carlos J. Bernardos, Antonio De Domenico, Marco Di Girolamo, Massinissa Lalam, Andreas Maeder, Dario Sabella, and Dirk Wübben. Cloud technologies for flexible 5g radio access networks. *IEEE Communications Magazine*, 52(5):68–76, 2014.

[8] Wind River. vran: The next step in network transformation. *White Paper*, 2017.

[9] Peter Rost, Ignacio Berberana, Andreas Maeder, Henning Paul, Vinay Suryaprakash, Matthew Valenti, Dirk Wübben, Armin Dekorsy, and Gerhard Fettweis. Benefits and challenges of virtualization in 5g radio access networks. *IEEE Communications Magazine*, 53(12):75–82, 2015.

[10] ORAN Alliance. O-ran: Towards an open and smart ran, white paper, October 2018.

[11] Michele Polese, Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges, 2022.

[12] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[13] Hatim Chergui, Luis Blanco, Luis A. Garrido, Kostas Ramantas, Sławomir Kukliński, Adlen Ksentini, and Christos Verikoukis. Zero-touch ai-driven distributed management for energy-efficient 6g massive network slicing. *IEEE Network*, 35(6):43–49, 2021.

[14] Shunliang Zhang. An overview of network slicing for 5g. *IEEE Wireless Communications*, 26(3):111–117, 2019.

[15] Sourjya Bhaumik, Shoban Preeth Chandrabose, Manjunath Kashyap Jataprolu, Gautam Kumar, Anand Muralidhar, Paul Polakos, Vikram Srinivasan, and Thomas Woo. Cloudiq: A framework for processing base stations in a data center. Mobicom '12, page 125–136.

[16] S. Niknam, A. Roy, SḊhillon, H. S. Singh, R. Banerji, H. Reed, J. N. Saxena, and S. Yoon. Intelligent o-ran for beyond 5g and 6g wireless networks. *arXiv preprint arXiv:2005.08374*, 2020.

[17] Jose A Ayala-Romero, Andres Garcia-Saavedra, Marco Gramaglia, Xavier Costa-Perez, Albert Banchs, and Juan J Alcaraz. vrain: Deep learning based orchestration for computing and radio resources in vrans. *IEEE Transactions on Mobile Computing*, 2020.

[18] Dario Bega, Albert Banchs, Marco Gramaglia, Xavier Costa-Pérez, and Peter Rost. Cares: Computation-aware scheduling in virtualized radio access networks. *IEEE Transactions on Wireless Communications*, 17(12), 2018.

[19] David M. Gutierrez-Estevez, Marco Gramaglia, Antonio de Domenico, Nicola di Pietro, Sina Khatibi, Kunjan Shah, Dimitris Tsolkas, Paul Arnold, and Pablo Serrano. The path towards resource elasticity for 5g network architecture. In *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 214–219, 2018.

[20] Krishna C. Garikipati, Kassem Fawaz, and Kang G. Shin. Rt-opex: Flexible scheduling for cloud-ran processing. CoNEXT '16, page 267–280. Association for Computing Machinery.

[21] Hatem Khedher, Sahar Hoteit, Patrick Brown, Ruby Krishnaswamy, William Diego, and Véronique Vèque. Processing time evaluation and prediction in cloud-ran. In *IEEE International Conference on Communications (ICC)*, pages 1–6, 2019.

[22] Po-Chiang Lin and Sheng-Lun Huang. Performance profiling of cloud radio access networks using openairinterface. In *IEEE Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 454–458, 2018.

[23] Tuyen X Tran, Ayman Younis, and Dario Pompili. Understanding the computational requirements of virtualized baseband units using a programmable cloud radio access network testbed. In *IEEE International Conference on Autonomic Computing (ICAC)*, pages 221–226. IEEE, 2017.

[24] Matthew C. Valenti, Salvatore Talarico, and Peter Rost. The role of computational outage in dense cloud-based centralized radio access networks. In *2014 IEEE Global Communications Conference*, pages 1466–1472.

[25] Andres Garcia-Saavedra, Xavier Costa-Perez, Douglas J Leith, and George Iosifidis. Fluidran: Optimized vran/mec orchestration. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 2366–2374, 2018.

[26] Sharda Tripathi, Corrado Puligheddu, Carla Fabiana Chiasserini, and Federico Mungari. A context-aware radio resource management in heterogeneous virtual rans. *IEEE Transactions on Cognitive Communications and Networking*, 2021.

[27] Gines Garcia-Aviles, Andres Garcia-Saavedra, Marco Gramaglia, Xavier Costa-Perez, Pablo Serrano, and Albert Banchs. Nuberu: Reliable ran virtualization in shared platforms. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, MobiCom '21, 2021.

[28] Ismael Gomez-Miguelez, Andres Garcia-Saavedra, Paul D. Sutton, Pablo Serrano, Cristina Cano, and Doug J. Leith. Srslte: An open-source platform for lte evolution and experimentation. In *WiNTECH*, page 25–32. Association for Computing Machinery, 2016.

[29] 3gpp ts 36.213 v12.4.0, release 12, evolved universal terrestrial radio access (e-utra) physical layer procedures. Technical report, 2014.

[30] S. Pramanik, A. Ksentini, and F. Chiasserini, C.˙ Characterizing the computational and memory requirements of virtual rans. In *2022 17th Wireless On-Demand Network Systems and Services Conference (WONS)*, pages 1–8, 2022.

[31] Sina Khatibi, Kunjan Shah, and Mustafa Roshdi. Modelling of computational resources for 5g ran. In *2018 European Conference on Networks and Communications (EuCNC)*, 2018.

[32] Peter Rost, Andreas Maeder, Matthew C. Valenti, and Salvatore Talarico. Computationally aware sum-rate optimal scheduling for centralized radio access networks. In *2015 IEEE Global Communications Conference (GLOBECOM)*, 2015.

[33] Peter Rost, Salvatore Talarico, and Matthew C. Valenti. The complexity–rate tradeoff of centralized radio access networks. *IEEE Transactions on Wireless Communications*, 14(11):6164–6176, 2015.

[34] Ke Wang, XiaoYi Yu, WenLiang Lin, ZhongLiang Deng, and Xin Liu. Computing aware scheduling in mobile edge computing system. *Wireless Networks*, 27(6):4229–4245, 2021.

[35] 3gpp ts 38.300 v16.8.0, "technical specification group radio access network; nr; nr and ng-ran overall description; stage 2 (release 16),. Technical report, Dec 2021.

[36] 5g america, "new services and applications with 5g ultra-reliable low latency communications," white paper,. Technical report, Nov. 2018.

[37] 3gpp r1-1700374, "downlink multiplexing of embb and urllc transmission,". Technical report, Jan. 2017.

[38] J. Huang and L. Gao. *Wireless Network Pricing*, volume 6. 2013.

[39] Anupam Kumar Bairagi, Md. Shirajum Munir, Madyan Alsenwi, Nguyen H. Tran, Sultan S. Alshamrani, Mehedi Masud, Zhu Han, and Choong Seon Hong. Coexistence mechanism between embb and urllc in 5g wireless networks. *IEEE Transactions on Communications*, 69(3):1736–1749, 2021.

[40] Arjun Anand, Gustavo De Veciana, and Sanjay Shakkottai. Joint scheduling of urllc and embb traffic in 5g wireless networks. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 1970–1978, 2018.

[41] Madyan Alsenwi, Nguyen H. Tran, Mehdi Bennis, Anupam Kumar Bairagi, and Choong Seon Hong. embb-urllc resource slicing: A risk-sensitive approach. *IEEE Communications Letters*, 23(4):740–743, 2019.

[42] Ali Esmaeily, H. V. Kalpanie Mendis, Toktam Mahmoodi, and Katina Kralevska. Beyond 5g resource slicing with mixed-numerologies for mission critical urllc and embb coexistence. *IEEE Open Journal of the Communications Society*, 4:727–747, 2023.

[43] Yerra Prathyusha and Tsang-Ling Sheu. Coordinated resource allocations for embb and urllc in 5g communication networks. *IEEE Transactions on Vehicular Technology*, 71(8):8717–8728, 2022.

[44] Yan Huang, Shaoran Li, Chengzhang Li, Y. Thomas Hou, and Wenjing Lou. A deep-reinforcement-learning-based approach to dynamic embb/urllc multiplexing in 5g nr. *IEEE Internet of Things Journal*, 7(7):6439–6456, 2020.

[45] Madyan Alsenwi, Nguyen H. Tran, Mehdi Bennis, Anupam Kumar Bairagi, and Choong Seon Hong. embb-urllc resource slicing: A risk-sensitive approach. *IEEE Communications Letters*, 23(4):740–743, 2019.

[46] Peng Yang, Xing Xi, Tony Q. S. Quek, Jingxuan Chen, Xianbin Cao, and Dapeng Wu. How should i orchestrate resources of my slices for bursty urllc service provision? *IEEE Transactions on Communications*, 69(2):1134–1146, 2021.

[47] Stefan Parkvall, Erik Dahlman, Anders Furuskar, and Mattias Frenne. Nr: The new 5g radio access technology. *IEEE Communications Standards Magazine*, 1(4):24–30, 2017.

[48] Wen Wu, Nan Chen, Conghao Zhou, Mushu Li, Xuemin Shen, Weihua Zhuang, and Xu Li. Dynamic ran slicing for service-oriented vehicular networks via constrained learning. *IEEE Journal on Selected Areas in Communications*, 39(7):2076–2089, 2021.

[49] Yuxiu Hua, Rongpeng Li, Zhifeng Zhao, Xianfu Chen, and Honggang Zhang. Gan-powered deep distributional reinforcement learning for resource management in network slicing. *IEEE Journal on Selected Areas in Communications*, 38(2):334–349, 2020.

[50] Somreeta Pramanik, Adlen Ksentini, and Carla Fabiana Chiasserini. Cost-efficient slicing in virtual radio access networks. *Computer Communications*, 209:349–358, 2023.

[51] Madyan Alsenwi, Nguyen H. Tran, Mehdi Bennis, Shashi Raj Pandey, Anupam Kumar Bairagi, and Choong Seon Hong. Intelligent resource slicing for embb and urllc coexistence in 5g and beyond: A deep reinforcement learning based approach. *IEEE Transactions on Wireless Communications*, 20(7):4585–4600, 2021.

[52] 3GPP. Study on new radio access technology physical layer aspects, 2017.

[53] Haojun Yang, Kan Zheng, Kuan Zhang, Jie Mei, and Yi Qian. Ultra-reliable and low-latency communications for connected vehicles: Challenges and solutions. *IEEE Network*, 34(3):92–100, 2020.

[54] Mehdi Setayesh, Shahab Bahrami, and Vincent W.S. Wong. Resource slicing for embb and urllc services in radio access network using hierarchical deep learning. *IEEE Transactions on Wireless Communications*, 2022.

[55] Sihem Bakri, Pantelis A. Frangoudis, Adlen Ksentini, and Maha Bouaziz. Data-driven ran slicing mechanisms for 5g and beyond. *IEEE Transactions on Network and Service Management*, 18(4):4654–4668, 2021.

[56] Leonard Kleinrock. *Theory, Volume 1, Queueing Systems*. Wiley-Interscience, USA, 1975.

[57] https://www.gurobi.com/.

[58] Klaus I. Pedersen, Guillermo Pocovi, Jens Steiner, and Saeed R. Khosravirad. Punctured scheduling for critical low latency data on a shared channel with mobile broadband. In *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, pages 1–6, 2017.

[59] Somreeta Pramanik, Adlen Ksentini, and Carla Fabiana Chiasserini. Cost-efficient slicing in virtual radio access networks. *Computer Communications*, 209:349–358, 2023.

[60] https://en.wikipedia.org/wiki/Divide-and-conquer_algorithm.

[61] Andres Garcia-Saavedra and Xavier Costa-Pérez. O-RAN: Disrupting the virtualized RAN ecosystem. *IEEE Communications Standards Magazine*, 5(4):96–103, 2021.

[62] Open RAN Alliance. O-RAN: Towards an Open and Smart RAN. *White Paper*, 2018.

[63] Cisco, Rakuten, Altiostar. Reimagining the End-to-End Mobile Network in the 5G Era. *White Paper*, 2019.

[64] Samsung. Virtualized Radio Access Network: Architecture, Key technologies and Benefits. *Technical Report*, 2019.

[65] Intel. vRAN: The Next Step in Network Transformation. *White Paper*, 2017.

[66] Gines Garcia-Aviles, Andres Garcia-Saavedra, Marco Gramaglia, Xavier Costa-Perez, Pablo Serrano, and Albert Banchs. Nuberu: Reliable ran virtualization in shared platforms. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, MobiCom '21, page 749–761, New York, NY, USA, 2021. Association for Computing Machinery.

[67] Nokia. The edge cloud: An agile foundation to support advanced new services. *White Paper*, 2018.

[68] Andres Garcia-Saavedra, George Iosifidis, Xavier Costa-Perez, and Douglas J Leith. Joint optimization of edge computing architectures and radio access networks. *IEEE Journal on Selected Areas in Communications*, 36(11):2433–2443, 2018.

[69] Jose A. Ayala-Romero et al. EdgeBOL: automating energy-savings for mobile edge AI. In *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, pages 397–410, 2021.

[70] Jose A. Ayala-Romero, Andres Garcia-Saavedra, Xavier Costa-Perez, and George Iosifidis. Bayesian online learning for energy-aware resource orchestration in virtualized RANs. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10, 2021.

[71] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

[72] Ismael Gomez-Miguelez, Andres Garcia-Saavedra, Paul D Sutton, Pablo Serrano, Cristina Cano, and Doug J Leith. srsLTE: An open-source platform for LTE evolution and experimentation. In *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, pages 25–32, 2016.

[73] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. Toward a practical perceptual video quality metric. *The Netflix Tech Blog*, 6(2), 2016.

[74] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 187–198, 2014.

[75] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998.

[76] Alexander A. Sherstov and Peter Stone. Function approximation via tile coding: Automating parameter choice. In Jean-Daniel Zucker and Lorenza Saitta, editors, *Abstraction, Reformulation and Approximation*, pages 194–205, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[77] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 66–83. Springer, 2017.

[78] S. Pasteris, S. Wang, M. Herbster, and T. He. Service placement with provable guarantees in heterogeneous edge computing systems. In *IEEE INFOCOM*, pages 514–522, 2019.

[79] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas. Joint service placement and request routing in multi-cell mobile edge computing networks. In *IEEE INFOCOM*, pages 10–18, 2019.

[80] X. Ma, A. Zhou, S. Zhang, and S. Wang. Cooperative service caching and workload scheduling in mobile edge computing. In *IEEE INFOCOM*, pages 2076–2085, 2020.

[81] F. Wang, C. Zhang, F. wang, J. Liu, Y. Zhu, H. Pang, and L. Sun. Intelligent edge-assisted crowdcast with deep reinforcement learning for personalized qoe. In *IEEE INFOCOM*, pages 910–918, 2019.

[82] Q. Liu, S. Huang, J. Opadere, and T. Han. An edge network orchestrator for mobile augmented reality. In *IEEE INFOCOM*, pages 756–764, 2018.

[83] C. Wang, S. Zhang, Y. Chen, Z. Qian, J. Wu, and M. Xiao. Joint configuration adaptation and bandwidth allocation for edge-based real-time video analytics. In *IEEE INFOCOM*, pages 257–266, 2020.

[84] Y. He, J. Ren, G. Yu, and Y. Cai. Optimizing the learning performance in mobile augmented reality systems with CNN. *IEEE Transactions on Wireless Communications*, 19(8):5333–5344, 2020.

[85] R. Combes and A. Proutiere. Dynamic rate and channel selection in cognitive radio systems. *IEEE J. Sel. Areas Commun.*, 33(5):910–921, May 2015.

[86] R. Combes, J. Ok, A. Proutiere, D. Yun, and Y. Yi. Optimal rate sampling in 802.11 systems: Theory, design, and implementation. *IEEE Trans. Mobile Comput.*, 18(5):1145–1158, May 2019.

[87] H. Gupta, A. Eryilmaz, and R. Srikant. Low-complexity, low-regret link rate selection in rapidly-varying wireless channels. In *Proc. IEEE Conf. Comput. Commun.*, pages 540–548, 2018.

[88] J. Ma, T. Nagatsuma, S. Kim, and M. Hasegawa. A machine-learning-based channel assignment algorithm for iot. In *Proc. Intl. Conf. Artificial Intell. Informat. Commun. (ICAIIC)*, pages 1–6, 2019.

[89] S. Hasegawa, S. Kim, Y. Shoji, and M. Hasegawa. Performance evaluation of machine learning based channel selection algorithm implemented on iot sensor devices in coexisting IoT networks. In *Proc. IEEE Consumer Commun. Netw. Conf. (CCNC)*, pages 1–5, 2020.

[90] M. A. Qureshi and C. Tekin. Fast learning for dynamic resource allocation in AI-enabled radio networks. *IEEE Trans. Cogn. Commun. Netw.*, 6(1):95–110, Mar. 2020.

[91] I. Comşa, S. Zhang, M. E. Aydin, P. Kuonen, Y. Lu, R. Trestian, and G. Ghinea. Towards 5G: A reinforcement learning-based scheduling solution for data traffic management. *IEEE Trans. Netw. Service Manag.*, 15(4):1661–1675, Dec. 2018.

[92] I. Comșa, R. Trestian, G. Muntean, and G. Ghinea. 5MART: A 5G SMART scheduling framework for optimizing QoS through reinforcement learning. *IEEE Trans. Netw. Service Manag.*, 17(2):1110–1124, June 2020.

[93] Sharda Tripathi, Corrado Puligheddu, Carla Fabiana Chiasserini, and Federico Mungari. A context-aware radio resource management in heterogeneous virtual rans. volume 8, pages 321–334, 2022.

[94] L. Zhang, J. Tan, Y. Liang, G. Feng, and D. Niyato. Deep reinforcement learning-based modulation and coding scheme selection in cognitive heterogeneous networks. *IEEE Trans. Wireless Commun.*, 18(6):3281–3294, Apr. 2019.

[95] X. Li, J. Fang, W. Cheng, H. Duan, Z. Chen, and H. Li. Intelligent power control for spectrum sharing in cognitive radios: A deep reinforcement learning approach. *IEEE Access*, 6:25463–25473, Apr. 2018.

[96] Y. S. Nasir and D. Guo. Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks. *IEEE J. Sel. Areas Commun.*, 37(10):2239–2250, Aug. 2019.

[97] Luca Baldesi, Francesco Restuccia, and Tommaso Melodia. Charm: Nextg spectrum sharing through data-driven real-time o-ran dynamic control. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, 2022.

[98] S. Tripathi, C. Puligheddu, S. Pramanik, A. Garcia-Saavedra, and C. F. Chiasserini. VERA: Resource orchestration for virtualized services at the edge. In *2022 IEEE International Conference on Communications (ICC)*, 2022.

[99] Yuanwei Liu, Shuowen Zhang, Xidong Mu, Zhiguo Ding, Robert Schober, Naofal Al-Dhahir, Ekram Hossain, and Xuemin Shen. Evolution of noma toward next generation multiple access (ngma) for 6g. *IEEE Journal on Selected Areas in Communications*, 40(4):1037–1071, 2022.

[100] Yuanwei Liu, Zhijin Qin, Maged Elkashlan, Zhiguo Ding, Arumugam Nallanathan, and Lajos Hanzo. Nonorthogonal multiple access for 5g and beyond. *Proceedings of the IEEE*, 105(12):2347–2381, 2017.

[101] Yuanwei Liu, Zhijin Qin, Maged Elkashlan, Arumugam Nallanathan, and Julie A. McCann. Non-orthogonal multiple access in large-scale heterogeneous networks. *IEEE Journal on Selected Areas in Communications*, 35(12):2667–2680, 2017.

[102] Hatim Chergui, Luis Blanco, Luis A. Garrido, Kostas Ramantas, Sławomir Kukliński, Adlen Ksentini, and Christos Verikoukis. Zero-touch ai-driven distributed management for energy-efficient 6g massive network slicing. *IEEE Network*, 35(6):43–49, 2021.

[103] O-RAN Working Group 2. Ai/ml workflow description and requirements-v01.03. Technical report, O-RAN Alliance, 2021.

[104] Emil Björnson and Luca Sanguinetti. Scalable cell-free massive mimo systems. *IEEE Transactions on Communications*, 68(7):4247–4261, 2020.

[105] O-RAN Working Group 1. Use cases analysis report r003-v12.00. Technical report, O-RAN Alliance, 2023.