Sharing Heterogeneous Computing Resources in Virtualized Open Radio Access Networks

by

Leonardo Lo Schiavo

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in

Telematic Engineering

Universidad Carlos III de Madrid

Advisors:
Dr. Marco Fiore
Dr. Andres Saavedra Garcia

Tutor:
Dr. Marco Fiore

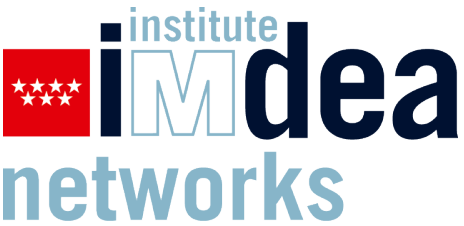
January 2025

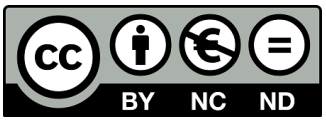
Sharing Heterogeneous Computing Resources in Virtualized Open Radio Access Networks

Prepared by:
Leonardo Lo Schiavo, IMDEA Networks Institute, Universidad Carlos III de Madrid
contact: leonardo.loschiavo@imdea.org, leonardo.loschiavo@alumnos.uc3m.es

Under the advice of:
Dr. Marco Fiore, IMDEA Networks Institute
Dr. Andres Saavedra Garcia, NEC Laboratories Europe GmbH

This work has been supported by:





"Y no tengo pensado hundirme aca tirado,
Y no tengo planeado morirme desangrado,
Y no me pidas que no vuelva a intentar,
Que las cosas vuelvan a su lugar."

Acknowledgements

This thesis represents the culmination of an intense and rewarding journey called PhD. Before presenting this work, I would like to take a moment and use this space to thank all the people who shared their time and support with me throughout this experience.

First of all, I would like to express my heartfelt gratitude and sincere appreciation to my PhD advisors, Dr. Marco Fiore, Dr. Andres Saavedra Garcia, and Dr. Albert Banchs for their expertise, guidance and continuous support. Working with you has been an incredibly meaningful experience for my professional growth and has greatly shaped the early stage of my research career and my scientific mindset. Thank you for your unconditional trust and patience during the many difficult moments of my PhD. I truly recognize that without any of you, achieving the results upon which this work builds would not have been possible, and I would not be the person and researcher I am today.

Next, I would like to thank my collaborators Dr. Marco Gramaglia, Dr. Jose A. Ayala-Romero, Dr. Gines Garcia-Aviles, and Dr. Xavier Costa-Perez for their collaboration and insightful feedback. Your contributions have been fundamental to our publications, and I truly believe that the results we achieved together are a testament to our teamwork.

I would also like to thank Prof. Tommaso Melodia, Dr. Michele Polese, Dr. Salvatore D'Oro, and Dr. Leonardo Bonati for hosting me for six months at Northeastern University in Boston. I have truly appreciated the opportunity to work in a top-tier research lab while living an unforgettable experience. It has been a formative professional and personal journey, allowing me to see research and life from a completely different perspective.

I would now like to express my genuine gratitude to Giulia, who was the first to encourage me to pursue a PhD at IMDEA. After five years, I want to thank you again for pushing me to make this choice and for the special friendship we have shared during our time in Madrid. Without your persistence, I doubt I would be here writing this thesis.

I would like to extend my gratitude to those colleagues who have become dear friends and have provided invaluable support during my PhD journey. Thanks to Francesco for your genuine friendship and for your many encouraging words–they have helped me immensely. Thanks to Sachit for your unwavering friendship and for standing by my side during the toughest moments of my professional journey throughout this PhD. Thanks to Livia for always being there to listen to my endless 'confessions', for your patience during

our coffee breaks, and for your countless personal and professional tips. Thanks Stavros for always believing in me even more than I did at times–your attitude and our synergy have provided a safe haven during the most challenging moments. Thanks Dayrene for your constant words of comfort, our endless laughter, and our salsa dance sessions.

Next, I would like to express my gratitude to all the friends and colleagues who supported me during these years, starting with my friends from the extended Italian community: in no particular order, thank you to Antonio, Pippo, Leo, Vittorio, Alessio, Valerio, Maurizio, Boris, Stefano, Luigi, Cami, Federica, Mer, Pali, Brenda, Marco, Andrea, Mauro, Laura, Diletta, Federico, Simona, and Marta. You have made these years unforgettable and filled with incredible experiences. I also want to extend my thanks to my colleagues in the Network Data Science group: Antonio, Diego, Michele, Nadezhda, Abhishek, Akem, Alan, Andre, Beyza, David, Erfan, Maximo, Orlando, and Sergi. Collaborating with such talented researchers has greatly enhanced my learning and encouraged me to always aim higher. I would also like to thank all the other colleagues I have met during my time at IMDEA Networks for the engaging conversations, lunches, and coffee breaks we shared. I would also like to extend my gratitude to my special colleagues at Northeastern University: Neagin, Manuel, Maria, Stefano, and Davide. Thank you for welcoming me so warmly during my experience in the United States.

Furthermore, I want to thank the friends of the Milan Club Madrid: Francesco, Fabio, Matteo, Grima, Riky, Federico, Luis, Cesar, and Pietro, as well as the friends from the Milan Club Boston: Simone, Joe, and Mark. You have brightened many weekends celebrating the successes of our Milan and created countless fun memories together.

Quindi, voglio ringraziare i miei amici di sempre in Italia: Michele, Giovanni, Simone, Dario, Davide, e Alberto. Nonostante la distanza, sapere di poter sempre contare su di voi mi ha aiutato tanto durante tutta la mia esperienza di dottorato.

Infine, voglio ringraziare la mia famiglia per essere incondizionatamente presente nei momenti difficili e per essere sempre orgogliosa di me e delle mie scelte, tra cui quella di intraprendere un percorso di dottorato. Un grazie infinito a mia madre, a mio padre, ai miei fratelli Giuseppe e Vincenzo, a mia cognata Francesca e alla mia nipotina Alba. Siete stati, siete e sarete sempre un importante punto di riferimento nella mia vita.

Un grazie va anche a me stesso, per non aver mai mollato quando le cose sembravano impossibili, e per le sensazioni provate prima e dopo quei 13 indimenticabili giorni di gloria che oggi mi permettono di scrivere questo elaborato e di dire di avercela fatta.

Published and Submitted Content

This thesis is based on the following published papers:

[1] **Leonardo Lo Schiavo**, Gines Garcia-Aviles, Andres Garcia-Saavedra, Marco Gramaglia, Marco Fiore, Albert Banchs, and Xavier Costa-Perez, "CloudRIC: Open Radio Access Network (O-RAN) Virtualization with Shared Heterogeneous Computing". In: *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking (ACM MobiCom 2024)*, November 18–22, 2024, Washington D.C., DC, USA., pp. 558–572, doi: https://doi.org/10.1145/3636534.3649381.

- This work is fully included and the contents are reported in Chapters 2, 3, and 6.
- The thesis author is the first author of this work and led the design of CloudRIC architecture and machine learning models, the collection of the decoding datasets and the experimental evaluation in software, and participated in writing parts of the manuscript.
- The material from this source included in this thesis is not singled out with typographic means and references.

[2] **Leonardo Lo Schiavo**, Jose A. Ayala-Romero, Andres Garcia-Saavedra, Marco Fiore, and Xavier Costa-Perez, "YinYangRAN: Resource Multiplexing in GPU-Accelerated Virtualized RANs". In: *Proceedings of the IEEE Conference on Computer Communications (IEEE INFOCOM 2024)*, May 20–23, 2024, Vancouver, BC, Canada, pp. 721–730, doi: https://doi.org/10.1109/INFOCOM52122.2024.10621380.

- This work is fully included and the contents are reported in Chapters 2 and 5.
- The thesis author is the first author of this work and led the implementation of YinYangRAN resource sharing mechanism, took part in the problem formulation, ran all the experiments of the evaluation, and participated in writing parts of the manuscript.
- The material from this source included in this thesis is not singled out with typographic means and references.

[3] Jose A. Ayala-Romero, **Leonardo Lo Schiavo**, Andres Garcia-Saavedra, and Xavier Costa-Perez, "Mean-Field Multi-Agent Contextual Bandit for Energy-Efficient Resource Allocation in vRANs". In: *Proceedings of the IEEE Conference on Computer*

Communications (IEEE INFOCOM 2024), May 20–23, 2024, Vancouver, BC, Canada, pp. 911-920, doi: https://doi.org/10.1109/INFOCOM52122.2024.10621197.

- This work is fully included and the contents are reported in Chapter 4.
- The thesis author is the second author of this work and led the collection of latency and energy decoding datasets, ran part of the experimental evaluation, and participated in writing parts of the manuscript.
- The material from this source included in this thesis is not singled out with typographic means and references.

[4] Leonardo Lo Schiavo, Gines Garcia-Aviles, Andres Garcia-Saavedra, Marco Gramaglia, Marco Fiore, Albert Banchs, and Xavier Costa-Perez, "CloudRIC demo: Open Radio Access Network (O-RAN) Virtualization with Shared Heterogeneous Computing". In: *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking (ACM MobiCom 2024)*, November 18–22, 2024, Washington D.C., DC, USA., 3 pages, doi: https://doi.org/10.1145/3636534.3698858.

- This demo is fully included and the contents are reported in Chapter 6.
- The thesis author led the design of CloudRIC framework and machine learning models which are demonstrated in this work, and participated in writing parts of the manuscript.
- The material from this source included in this thesis is not singled out with typographic means and references.

Other publications that were produced during the doctoral thesis but are not included in this manuscript include:

[5] Leonardo Lo Schiavo, Marco Fiore, Marco Gramaglia, Albert Banchs, and Xavier Costa-Perez, "Forecasting for Network Management with Joint Statistical Modelling and Machine Learning". In: *Proceedings of the 23rd IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (IEEE WoWMoM 2022)*, June 14–17, 2022, Belfast, United Kingdom, pp. 60-69, doi: https://doi.org/10.1109/WoWMoM54355.2022.00028.

Abstract

Virtualization has recently become a fundamental paradigm in the implementation of 5G Networks, specifically in the context of Radio Access Networks (RANs). RAN virtualization enables baseband processing on general-purpose computing platforms, thus overcoming the coupling of RAN functions with dedicated hardware of traditional hardwired RANs. This approach allows RAN operators to disrupt traditional hardware vendor lock-ins and enable sharing and multiplexing of the available computing resources, as the RAN Base Station (BS) is disaggregated into minimal Radio Unit (RU) hardware connected to cloud-oriented computing platforms that run in software virtual signal processing tasks of the Distributed Unit (DU) and Centralized Unit (CU).

However, the execution of such tasks in a timely manner at high probabilities (*reliably)* is challenging due to their computationally intensive nature, especially at DU level. For this purpose, carrier-grade virtualized RANs (vRANs) today rely on general-purpose computing platforms equipped with Hardware Accelerators (HAs), which are generally energy-hungry and monetarily expensive but can guarantee DU processing reliability. Traditional HAs include Application-Specific Integrated Circuits (ASICs) and Field Programmable Gate Arrays (FPGAs). Recently, Graphics Processing Units (GPUs) have also been considered as HAs, hinging on their unique capability to be easily programmable in software and to efficiently process Machine Learning (ML) workloads, whose algorithms can be used to automate and optimize DU operations.

HAs also significantly increase the energy and monetary costs of vRANs, which puts at stake the environmental and economic sustainability of next-generation mobile networks. Therefore, deploying reliable hardware accelerated vRANs with the lowest possible energy toll while reducing deployment costs has become a challenging problem for operators. Indeed, current industrial solutions fail to provide energy and cost efficiency in vRAN, as respectively (i) more energy-efficient processors are shunned for DU tasks processing and (ii) dedicated HAs are assigned to individual DUs following an overprovisioned approach.

This thesis investigates the deployment of energy- and cost-effective yet reliable vRANs in order to close the gap with respect to the aforementioned standard solutions. The vision set forth by this thesis is twofold: (i) increasing the energy-efficiency of traditional hardware accelerated vRANs by opportunistically complementing HAs with

less energy-hungry computing processors like Central Processing Units (CPUs) to process DU tasks; (*ii*) improving the cost-efficiency by means of DU centralization, that allows to amortize the cost of each expensive HA by sharing the same HA across multiple DUs.

In line with this vision, the first solution proposed in this thesis is ECORAN, an efficient multi-agent contextual bandit ML algorithm operating in the O-RAN Near-Real-Time RAN Intelligent Controller (Near-RT-RIC). ECORAN configures policies to opportunistically offload DU workloads to either a GPU-based HA or CPUs in the O-Cloud to save energy while preserving the reliability of the vRAN. To address cost-efficiency, instead, ECORAN applies concepts from mean field theory to be fully scalable and thus deal with an arbitrarily large and dynamic number of DUs, that are centralized in the same shared and HA-powered computing platform. Using traffic traces from a production mobile network, ECORAN can provide up to 40% energy savings and roughly up to 60x cost gains with respect to the standard approach used today by the industry.

As with many other studies on RAN control in the literature, the offloading policy used in ECORAN is determined by an ML model that requires GPU resources for efficient training and execution in production. However, dedicating a GPU to each ML model that automates a specific RAN control function and reserving a GPU-based HA solely for DU processing is not a cost-efficient approach. Conversely, indiscriminately co-locating DU workloads and multiple ML services can compromise the processing reliability of the former and the throughput performance of the latter. Under this perspective, this thesis explores reliable multiplexing opportunities of the resources of a single GPU to further squeeze cost-efficiency in vRANs. To this end, this thesis proposes YinYangRAN, an innovative system operating in the Non-Real-Time-RIC that supervises the multiplexing of the computing resources of a GPU-based HA as to ensure reliability in processing DU tasks while maximizing the throughput of a concurrent ML service running in the same GPU. Experiments performed with workloads collected in real RANs show that YinYangRAN can potentially reduce the deployment cost by a factor of N compared to the solution using N dedicated GPUs per process, and improve vRAN reliability by over 50% compared to hardware-accelerated vRANs using conventional GPU multiplexing methods, with minimal impact on co-located ML workloads.

Based on insights from tracking workload dynamics in real-world cells, it is observed that traffic exhibits burstiness at the Transmission Time Interval (TTI) level, i.e. with a timescale of 1 ms. However, YinYangRAN operates in Non-Real-Time (timescale$\geq$ 1 s), and ECORAN operates in Near-Real-Time (timescale $\sim 10 - 100$ ms). Consequently, to ensure reliability in the vRAN, both solutions adopt a conservative approach by configuring resources for the highest expected peak over the entire decision period, that is significantly longer than 1 ms. This approach leads to wasting resources for most of the decision period due to overprovisioning. Thus, additional gains in energy and cost efficiency can potentially be achieved by exploiting a heterogeneous O-Cloud

infrastructure with both HAs and CPUs through a real-time controller capable of responding to TTI-level traffic fluctuations. To address this, this thesis introduces CloudRIC, a real-time brokering system powered by lightweight data-driven models that jointly coordinates a centralized access for multiple DUs to a heterogeneous pool of computing processors, including HAs and CPUs, and assists DUs with compute-aware radio policies while meeting vRAN-specific reliability targets. Extensive experimental evaluations on GPU-accelerated vRANs demonstrate that CloudRIC can achieve, respectively, 3x and 15x average gains in energy and cost efficiency under real and even dense RAN workloads compared to the industry-standard solution that assigns dedicated HAs to individual DUs, while maintaining the same 99.999% target reliability.

At the time of writing this thesis, the proposed solutions are, to the best of our knowledge, the only approaches aimed at reliably deploying energy- and cost-efficient hardware-accelerated vRANs through both DU centralization and the combined use of HAs and CPUs for DU processing.

Table of Contents

List of Tables

List of Figures

List of Acronyms

3GPP Third Generation Partnership Project

AAL Acceleration Abstraction Layer

AI Artificial Intelligence

ASIC Application-Specific Integrated Circuits

AWGN Additive White Gaussian Noise

BBdev Baseband Device Library

BS Base Station

CAGR Compound Annual Growth Rate

CB Code Blocks

CDF Cumulative Distribution Function

CEC Certainty Equivalent Control

COTS Commercial Off-The-Shelf

CP Control Plane

CPU Central Processing Units

CRC Cyclic Redundancy Check

CU Centralized Unit

CUDA Compute Unified Device Architecture

DDPG Deep Deterministic Policy Gradient

DL Deep Learning

DPDK Data Plane Development Kit

DRL Deep Reinforcement Learning

DU Distributed Unit

EAL Environment Abstraction Layer

eCPRI Evolved Common Public Radio Interface

FEC Forward Error Correction

FIFO First In First Out

FPGA Field Programmable Gate Arrays

FH Fronthaul Network

GPU Graphics Processing Units

HA Hardware Accelerators

HP High Performance

IMS Infrastructure Management Service

LDPC Low-Density Parity-Check

LLR Log-Likelihood Ratios

LPU Logical Processing Units

MAC Medium Access Control

MAE Mean Absolute Error

MARL Multi-Agent Reinforcement Learning

MCFS Mixed-Criticality Federated Scheduling

MCS Modulation and Coding Scheme

MDP Markov Decision Process

MIG Multi-Instance GPU

MIMO Multiple-Input Multiple-Output

ML Machine Learning

MPS Multi-Process Sharing

MWT Minimum Waiting Time

NF Network Functions

NFV Network Function Virtualization

NIC Network Interface Card

NN Neural Networks

Non-RT Non-Real-Time

NR New Radio

NRE Non-Recurrent-Engineering

Near-RT Near-Real-Time

OFDM Orthogonal Frequency-Division Multiplexing

ONNX Open Neural Network Exchange

OS Operating System

PCI Peripheral Component Interconnect

PCIe Peripheral Component Interconnect Express

PHY Physical layer

QoS Quality of Service

RAM Random Access Memory

RAN Radio Access Network

RB Resource Block

RF Radio Frequency

RIC RAN Intelligent Controller

RL Reinforcement Learning

RT Real-Time

RU Radio Unit

SDN Software Defined Networking

SIMD Single Instruction Multiple Data

SM Streaming Multiprocessors

SMO Service & Management Orchestrator

SNR Signal-to-Noise Ratio

TB Transport Block

TTI Transmission Time Interval

TTM Time-To-Market

UAV Unmanned Aerial Vehicles

UE User Equipment

UP User Plane

URLLC Ultra-Reliable Low-Latency Communication

VNF Virtual Network Functions

vRAN virtualized Radio Access Network

WCET Worst-Case Estimation Time

1 Introduction

The recent adoption of technical enablers like Software Defined Networking (SDN) and Network Function Virtualization (NFV) in 5G networks is making possible the shift from hardware-based to software-based network architectures, as network functions that were previously running in specific-purpose hardware are now being implemented as Virtual Network Functions (VNF) in software and running on general-purpose Cloud platforms.

Among the concepts building on those enablers, in recent years virtualized Radio Access Network (vRAN) has garnered considerable interest from the mobile telecommunications industry thanks to its many advantages over the traditional hardwired Radio Access Network (RAN). In particular, the capability of vRAN of processing Base Station (BS) functions on Commercial Off-The-Shelf (COTS) computing platforms in the edge cloud brings unprecedented flexibility in the RAN ecosystem, potentially disrupts the traditional hardware vendor lock-ins that have historically restrained innovation and competition, and breeds a new market with unique business prospects [6]. Indeed, under the lead of the O-RAN Alliance [7], all the major industry players are investing in 5G vRANs development [8, 9], and analysts forecast that vRAN market will outgrow the conventional RAN market and generate up to $29 billion in revenue by 2028 [10].

In 5G vRANs, the BS is no longer a monolithic proprietary hardware entity and is disaggregated into a minimal Radio Unit (RU) hardware, responsible for basic Radio Frequency (RF) operations, the Distributed Unit (DU), which performs Physical layer (PHY) and Medium Access Control (MAC) layer tasks, and the Centralized Unit (CU), in charge of the highest BS layers. The RU hardware is connected to general-purpose servers that host virtualized computing platforms, typically containers, implementing most of the DU and CU signal processing tasks (see Figure 1.1). Among these tasks, PHY tasks running at the DU are the most compute-intensive, such as Forward Error Correction (FEC): indeed, they must meet very stringent computing latency constraints as they contribute to a signal processing pipeline with hard deadlines in the 1-3 ms range [11–13], that must be met with 99.999% (*5-nines*) probability to guarantee *reliability* [12] and avoid that users drop connectivity after losing synchronization [13].

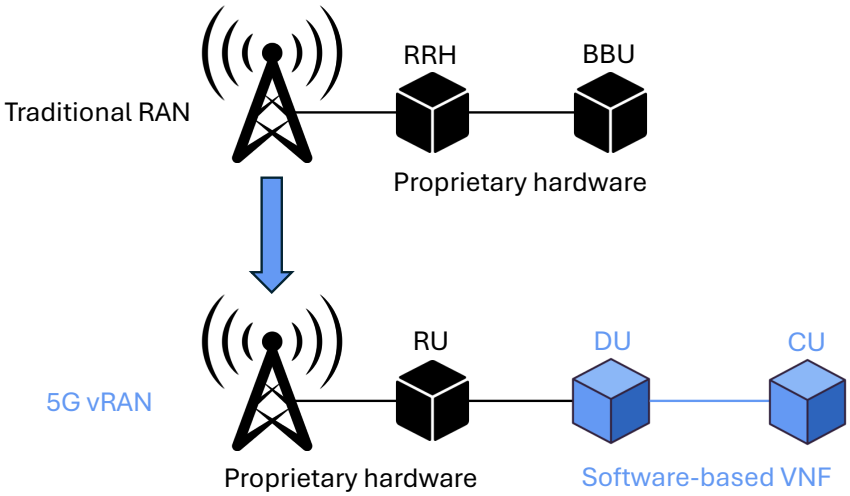


Figure 1.1: Comparison between traditional RAN and 5G vRAN.

1.1. Motivation

The traditional virtualization paradigm based on executing DU tasks as software functions running in general-purpose Central Processing Units (CPU) is alone not sufficient to meet the latency constraints with the target reliability in industry-grade scenarios, since these processors are generally too slow for the purpose.

To obtain the required latency gains over CPUs, the current approach in industry consists of offloading these tasks to dedicated Hardware Accelerators (HA) [12], such as Peripheral Component Interconnect Express (PCIe) integrating Application-Specific Integrated Circuits (ASIC) [14] or Field Programmable Gate Arrays (FPGA) [15].

Recently, also Graphics Processing Units (GPU) have proven to be a viable alternative as HAs, with commercial solutions developed by important market players like NVIDIA [16, 17]. Indeed, GPUs can provide latency guarantees to DU tasks by leveraging their massive computational parallelism offered by arrays of Streaming Multiprocessors (SM) [18], with the advantage of potentially co-locate GPU-based Machine Learning (ML) workloads that can perform automated fast decision-making to improve DU operations: examples include traffic load forecasting at RU level [19], ML-based network resource orchestration [20], policing of radio resource schedulers [21], policing of compute-aware radio schedulers [13, 22]. This unique capability of the GPU of potentially multiplexing its resources to both BS signal processing tasks and ML workloads is the major selling point by those operators that push for the wide adoption of this HA.

Figure 1.2 shows an example of a Cloud-based vRAN system: the hardware of the RU is connected to a COTS server in the edge cloud that uses either CPUs, GPUs, FPGAs or ASICs to execute in software the VNFs of the DU and CU.

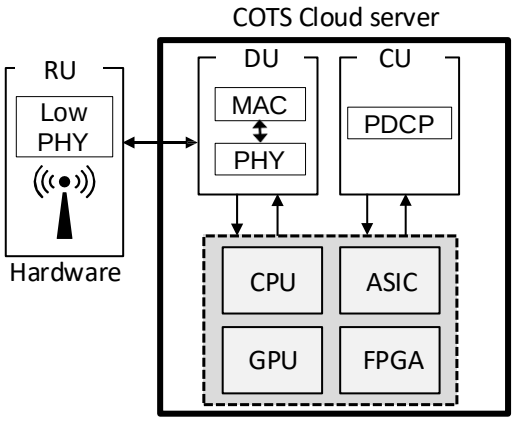


Figure 1.2: Cloud-based vRAN system.

However, deploying vRANs only with DU-dedicated HAs is casting doubts in the industry due to the economic and energetic costs of the HAs, as implied by top figures of Nokia [23], Ericsson [24] or Mavenir [25]. Indeed, when compared to CPU cores, GPUs are more expensive in terms of unit cost and ASICs show consistently higher Non-Recurrent-Engineering (NRE) costs and Time-To-Market (TTM) months [26], with the disadvantage of not being programmable. Table 1.1 reports these economic figures considering an Intel Xeon 6240R CPU, an NVIDIA V100 GPU, an Intel PAC N3000 FPGA and an Intel ACC100 ASIC.

	CPU Core	**GPU**	**FPGA**	**ASIC**
Programmable	Software	Software	Yes	No
TTM (months)	<2	<2	2	30
NRE ($)	0	0	0	350K-1000K
Unit cost ($)	110	8000	4000	3000

Table 1.1: Comparison of processors for vRANs.

Additionally, HAs are also energy-hungry processors, as for instance an Intel ACC100 ASIC and an NVIDIA V100 GPU can consume up to 52W and 250W respectively [14,27], which is 20-82% of the overall consumption of a commodity server [28]. In contrast, CPUs are generally more energy-efficient than HAs, but alone they are not capable of meeting the latency requirements of the DU tasks with the target 5-nines probability of industry-grade vRANs [29], despite Single Instruction Multiple Data (SIMD) programming and other optimizations [30, 31]. The trade-off between latency and energy consumption involving CPUs and HAs is better shown in Figure 1.3, illustrating the mean and max-min range performance of state-of-the-art libraries for DU's FEC Low-Density Parity-Check (LDPC) decoding function to process an LDPC-encoded Transport Block (TB): a CPU core is capable of processing reliably, i.e. within a typical deadline of 1-ms, only TBs below 100 Kb, while consuming less energy than a HA (GPU) within this range.

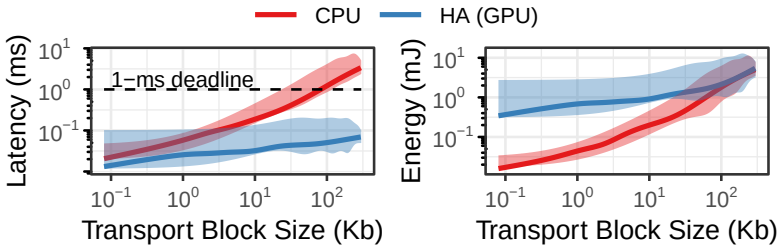


Figure 1.3: Performance of FEC LDPC decoding in CPU and HA (GPU).

1.2. Research challenges

As discussed in Section 1.1, the current industrial approach for deploying vRANs relies on DU-dedicated HAs, as only these processors are capable of ensuring the 5-nines processing latency probability, though at significant economic and energetic costs.

This thesis focuses on the deployment of hardware-accelerated vRANs at lower energy toll and deployment costs with respect to the standard industrial approach. However, this efficient deployment conveys several issues that need to be considered at the time of design, taking also into account the strict reliability requirements of industry-grade vRANs.

More details about the research challenges of this effective design are described next.

1.2.1. Energy efficiency in hardware-accelerated vRANs

To address the problem of energy efficiency in hardware-accelerated vRANs, this thesis demonstrates that CPUs are capable of handling some of DU workloads without the assistance of HAs by exploiting SIMD programming and other optimizations [30, 31]. Yet, CPUs alone cannot ensure 5-nines reliability for *all* workloads and, consequently, they are usually shunned for this job in industry-grade RANs [29]. Nevertheless, CPUs are a valuable complement to HAs in those tasks and balancing DU workloads between CPUs and HAs substantially improves the energy-efficiency of vRANs. The rationale is that minimizing processing latency brings no benefit as long as deadlines are met, hence CPUs may be occasionally exploited to alleviate the HAs' energy toll. Indeed, as illustrated in Figure 1.3 considering the FEC LDPC decoding task, a CPU core can decode within 1 ms TBs below 100 Kb consuming ∼5.7× less energy than a GPU-based HA. Although the O-RAN standard provides convenient abstractions for heterogeneous processors in vRANs including HAs and CPUs, it falls short to support DU with strategies at any timescale to attain such reliable and energy-efficient *opportunistic HA offloading*. Thus, the first research challenge of this thesis is to develop control strategies that assist DU processing with reliability and energy efficiency by leveraging CPUs along with HAs.

1.2.2. Cost efficiency in hardware-accelerated vRANs

To tackle the challenge of cost efficiency, instead, this thesis shows that the industrial approach of assigning individual DUs with dedicated individual HAs is an over-dimensioned solution, as the HAs suffer from low usage under real workloads. Therefore, maximizing the utilization of these expensive processors by reusing their available spare resources is key to amortizing their economic cost, illustrated in Table 1.1. In line with this vision, sharing individual HAs among multiple DUs by means of *DU centralization* or co-locating with DU processing other HA-based workloads would provide a more cost-effective vRAN deployment with respect to the HA-dedicated standard approach.

While the concept of DU centralization is not new, as 71% of operators in the United States intend to realize DU centralization solutions by 2025 [32], and some already implement it [33], the traditional RAN centralization approaches *only* exploit long-term traffic variations, such as day-night ones [34, 35], which is insufficient for cost-efficient RAN virtualization. Indeed, designing a solution that is efficiently and reliably pooling computing resources is technically challenging as it would require anticipatory operation that effectively copes with fluctuations and peaks in the future user demand without compromising the target reliability. Thus, the second research challenge of this thesis is to provide reliable coordination among multiple DUs that share the same HA.

To further amortize the cost of hardware-accelerated vRANs, the utilization of the HAs can be maximized by *multiplexing* their resources between DU workloads and other services that specifically need the HA to be processed efficiently. However, typical HAs in the industry, such as ASICs and FPGAs, are not programmable in software, which limits their ability to be easily multiplexed. For this reason, recently GPUs have been adopted as HAs thanks to their unique capability of multiplexing their resources between DU tasks and ML models. However, multiplexing GPU resources between *inelastic* DU workloads and *elastic* ML algorithms by ensuring processing latency targets of the former while maximizing the throughput of the latter requires unconventional resource control mechanisms that are not currently provided by state-of-the-art O-RAN specifications.

1.3. Contributions

In response to the questions raised in Section 1.2 regarding the deployment of energy- and cost-efficient yet reliable vRANs, this thesis has made several contributions resulting in peer-reviewed publications and open-source code. One publication has appeared in the proceedings of the *ACM MOBICOM 2024*, and other two have appeared in the proceedings of *IEEE INFOCOM 2024*. Both conferences are A* as ranked by the CORE2023[1] database. Other publications include one demo at *ACM MOBICOM 2024*.

[1] https://portal.core.edu.au/conf-ranks/2074/

The contributions of the thesis are as follows.

Contribution 1. *Characterization of CPU- and GPU-based DU PHY processors*
With the first contribution, CPU- and GPU-based processors for DU PHY tasks are characterized in terms of latency and power consumption using real-world mobile traffic data over an implementation of the standard-defined O-RAN Acceleration Abstraction Layer (AAL), making the dataset publicly available. GPUs are chosen over other HAs as they represent a new and still little understood 5G HA, with features like AI-on-5G and programmability [36] that are attractive for mobile operators [17]. To our knowledge, this is the first in-depth analysis of this emerging resource for 5G DU PHY processing.

The main findings of this analysis are: (*i*) SIMD-capable CPUs can handle 5G DU PHY workloads reliably in a wide range of contexts, which motivates our opportunistic HA offloading model for energy-efficient vRAN deployments; (*ii*) dedicating HAs to individual DUs results in dramatic under-utilization of these expensive resources with real-world workloads, which supports the strategy of DU centralization to achieve cost efficiency.

Contribution 2. *Energy- and cost-efficient vRANs with Near-Real-Time control*
Leveraging the findings of Contribution 1, the second contribution of this thesis is ECORAN [3], an energy-aware learning strategy that opportunistically offloads DUs compute-intensive workloads to HAs or CPUs to reduce the energy footprint of vRANs. This solution includes two elements: (*i*) a simple and fast threshold-based offloading rule (ECORAN-R) to operate in real-time ($<$ 1 ms); (*ii*) a multi-agent contextual bandit algorithm (ECORAN-P) to optimally configure the offloading rule for each DU (agent) in Near-Real-Time (Near-RT) ($\sim$100 ms). To reduce the economic impact of the usage of HAs in the vRAN, ECORAN applies concepts from mean field theory to deal with an arbitrarily large number of DUs, that are centralized in the same shared and HA-powered computing platform. Using traces collected from a real operational RAN, ECORAN provides significant energy savings and cost gains with respect to the industry standard approach of using dedicated HAs for each DU.

- Jose A. Ayala-Romero, **Leonardo Lo Schiavo**, Andres Garcia-Saavedra, and Xavier Costa-Perez, "Mean-Field Multi-Agent Contextual Bandit for Energy-Efficient Resource Allocation in vRANs". In: *Proceedings of the IEEE Conference on Computer Communications (IEEE INFOCOM 2024)*, May 20–23, 2024, Vancouver, BC, Canada, pp. 911-920, doi: https://doi.org/10.1109/INFOCOM52122.2024.10621197.

Contribution 3. *Cost-efficient GPU-accelerated vRANs*
To further enhance the overall efficiency of costly GPU-accelerated vRANs, this thesis devises YinYangRAN [2], a resource control mechanism designed to be integrated into the O-RAN's Service & Management Orchestrator (SMO) to dynamically allocate the minimum necessary GPU resources to 5G DUs to ensure maximal reliability, which

enables reusing spare resources for other co-located ML services. The solution is rooted in a two-stage approach derived from an approximate dynamic programming technique known as Certainty Equivalent Control (CEC). Using an experimental prototype and data from operational RANs, we demonstrate that YinYangRAN achieves lower deployment costs than the standard approach using dedicated HAs and consistently higher processing reliability than conventional GPU multiplexing models with minimal impact on co-located ML workloads. This is the first work that identifies and addresses the complex problem of HA management for cost efficiency in emerging GPU-acccelerated vRANs.

- **Leonardo Lo Schiavo**, Jose A. Ayala-Romero, Andres Garcia-Saavedra, Marco Fiore, and Xavier Costa-Perez, "YinYangRAN: Resource Multiplexing in GPU-Accelerated Virtualized RANs". In: *Proceedings of the IEEE Conference on Computer Communications (IEEE INFOCOM 2024)*, May 20–23, 2024, Vancouver, BC, Canada, pp. 721-730, doi: https://doi.org/10.1109/INFOCOM52122.2024.10621380.

Contribution 4. *Energy- and cost-efficient vRANs with Real-Time control*
To provide additional gains in energy and cost efficiency with respect to Contribution 2 based on Near-RT control, this thesis proposes CloudRIC [1], a solution that implements a Real-Time (RT) controller capable of promptly responding to sub-millisecond traffic fluctuations of real world RAN workloads. CloudRIC leverages a brokering system that jointly controls centralized access to shared resources of heterogeneous processors and optimizes radio scheduling policies across multiple DUs in RT. It seamlessly integrates into the O-RAN standard, and comprises two key elements: (*i*) an AAL Broker that maximizes energy efficiency by exploiting CPUs and performing HA offloading opportunistically in RT; (*ii*) a RT RAN Intelligent Controller (RIC) that provides compute-aware radio policies to ensure reliability in cost-efficient computing platforms shared among multiple DUs. A prototype of CloudRIC based on Data Plane Development Kit (DPDK)'s Environment Abstraction Layer (EAL) and Open Neural Network Exchange (ONNX) is comprehensively evaluated with realistic RAN mobile traffic demands. Experimental results demonstrate that CloudRIC achieves consistent average gains in energy and cost efficiency over the industry standard approach for vRANs, while incurring low implementation overheads and meeting the targeted 99.999% processing reliability even in severely congested scenarios.

- **Leonardo Lo Schiavo**, Gines Garcia-Aviles, Andres Garcia-Saavedra, Marco Gramaglia, Marco Fiore, Albert Banchs, and Xavier Costa-Perez, "CloudRIC: Open Radio Access Network (O-RAN) Virtualization with Shared Heterogeneous Computing". In: *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking (ACM MobiCom 2024)*, November 18–22, 2024, Washington D.C., DC, USA., pp. 558–572, doi: https://doi.org/10.1145/3636534.3649381.

The publications above are accompanied by publicly available material uploaded on Zenodo and GitHub. The list of the released material is presented in Table 1.2.

Material	URL	Paper
5G DU PHY CPU/GPU dataset	zenodo.org/records/10691661	[1]
CloudRIC AAL-B models weights	zenodo.org/records/10705309	[1]
ECORAN source code	github.com/jaayala/ECORAN	[3]

Table 1.2: Publicly available thesis material.

1.4. Outline of the thesis

The rest of the thesis is organized into different chapters that contextualize and describe in detail the solutions proposed in this thesis. The chapters are described next.

Chapter 2 establishes the groundwork of the thesis by introducing the background. At the beginning, a primer on 5G New Radio (NR) is presented, followed by a brief description of state-of-the-art hardware accelerator models. The chapter goes further to provide a background on modern techniques to perform resource multiplexing in GPU-based HAs and on RAN virtualization by describing the standard O-RAN architecture and Control Plane (CP). The final part of the chapter focuses on the description of existing works on O-RAN control and virtualization, energy efficiency in mobile networks, multi-agent learning for mobile networks, task scheduling, and HA resource multiplexing by exposing their limitations to cope with the proposed research challenges of this thesis.

Chapter 3 provides an experimental analysis to justify the main design choices proposed in this thesis to achieve energy and cost efficiency. In particular, the analysis on the reliability of CPUs and HAs when processing 5G DU PHY FEC decoding tasks to justify the complementarity of these two processors and the possibility of performing *opportunistic HA offloading* as a way to mitigate HAs energy toll. Then, the chapter analyses current utilization figures of GPU-based HAs to validate the opportunity of sharing individual HAs among multiple DUs or reusing spare HA resources with other co-located services with the aim of enhancing vRAN cost efficiency.

Chapter 4 draws on the findings of Chapter 3 to introduce ECORAN, a solution integrated into a real and standard-compliant O-RAN platform. ECORAN utilizes the Near-RT RIC to implement energy-efficient offloading rules, allowing multiple centrally-controlled DUs to opportunistically use HAs or CPUs for processing PHY workloads.

Chapter 5 focuses on the specific case of GPU-accelerated vRANs to further improve cost efficiency through GPU resource multiplexing. After assessing practical trade-offs and overheads of this mechanism, the chapter presents YinYangRAN, an innovative O-RAN-compliant solution that multiplexes GPU computing resources as to ensure 5G DU

PHY processing reliability while maximizing the throughput of co-located ML services.

Chapter 6 expands the analysis of Chapter 3, showing that real-world RAN traffic demands fluctuate at sub-millisecond timescale and that standard O-RAN policies are not viable to ensure reliability under this scenario. Leveraging this expanded analysis, the chapter presents CloudRIC, a system that is capable of meeting specific reliability targets in a cost- and energy-effective manner using lightweight data-driven models. To reach this goal, CloudRIC coordinates the access of multiple DUs to a heterogeneous set of computing processors and assists DUs with RT compute-aware radio scheduling policies.

Chapter 7 draws the most important findings and conclusions of the thesis by laying out possible directions of this research under short-term and long-term perspectives.

2 Background

The following chapter presents the building blocks of this thesis, based on the hardware acceleration of 5G New Radio (NR) data processing pipeline under the standardized and open O-RAN architecture for virtualized Radio Access Network (vRAN). Current state-of-the-art hardware acceleration models are described, followed by the introduction of modern techniques to multiplex the resources of one of the most common Hardware Accelerators (HA), the Graphics Processing Units (GPU)-based HA.

Then, existing works on vRAN resource optimization and orchestration, energy efficiency and multi-agent learning in mobile networks, task scheduling and HA resource multiplexing are examined, highlighting their limitations in addressing the research challenges introduced in Section 1.2.

2.1. 5G New Radio

5G base stations comprise a Radio Unit (RU), which performs basic radio operations such as signal sampling; a Centralized Unit (CU), which processes the highest layers; and a Distributed Unit (DU), which processes the Physical layer (PHY), Medium Access Control (MAC), and radio link control layers [37].

NR is 5G's PHY/MAC interface. Our focus is on sub-6GHz bands, which allow up to 100 MHz per carrier and have flexible *numerology* $\mu = \{0,1,2\}$ [38]. The basic spectrum unit is the Resource Block (RB), which encompasses 12 subcarriers with $15 \cdot 2^{\mu}$-KHz spacing. Time is divided into 1-ms subframes, each carrying 2^{μ} slots with, usually, 14 Orthogonal Frequency-Division Multiplexing (OFDM) symbols lasting $66.7 \cdot 2^{-\mu}$ μs. Every Transmission Time Interval (TTI), often one slot, the DU's MAC schedules one Transport Block (TB) for/from every active User Equipment (UE), which are signalled to UEs by *grants*. The TB size depends on the numerology, the amount of buffered data, the DU's RB scheduling policy, and the Modulation and Coding Scheme (MCS), selected based on the Signal-to-Noise Ratio (SNR). Uplink TBs must be sent within $K2$ slots of receiving the grant [39]. Figure 2.1 shows the pipeline of DU operations required to process a TB.

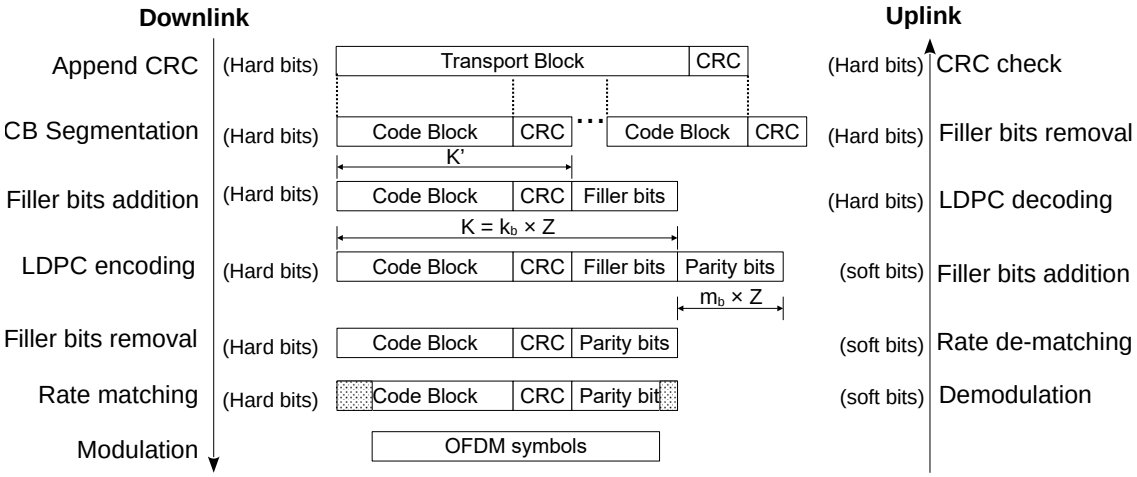


Figure 2.1: O-RAN DU data processing pipeline.

At the transmitter side, TBs are divided into Code Blocks (CB) with individual Cyclic Redundancy Check (CRC) fields. Filler bits adapt the CB size to the requirements of the Low-Density Parity-Check (LDPC) encoder used for Forward Error Correction (FEC), which produces a codeword with parity bits. Finally, the codeword is aligned to the capacity of the allocated RBs (which depends on their MCS) via rate matching, by applying puncturing or repetition. At the receiver side, a soft-output detector computes the reliability of the data as Log-Likelihood Ratios (LLR) called *soft bits*. Then, an LDPC decoder maps soft bits into hard bits through an iterative belief propagation algorithm. The algorithm terminates after a maximum number of iterations (usually 10), or earlier if CRC validates the codeword. The TB is reconstructed once all of its CBs are successfully decoded. More details can be found in [40].

To adhere to Third Generation Partnership Project (3GPP) and O-RAN requirements [11,37], processing the heavier LDPC tasks has a deadline $D = \{1, \ldots, 3\}$ ms, depending on the base station, which must be met with 99.999% probability to reach the industry's *5-nines* reliability target [12]. This is achieved today with DU-dedicated HAs.

2.2. Hardware acceleration

There are two hardware acceleration models, typically implemented with Application-Specific Integrated Circuits (ASIC), Field Programmable Gate Arrays (FPGA) or GPUs [29]: *in-line*, in which the HA directly processes the pipeline of Figure 2.1 as the radio wireless symbols arrive from the Network Interface Card (NIC), without software intervention; and *look-aside*, in which the HA operates on data managed by a software controller to perform only selected tasks, like for example LDPC decoding.

Traditionally, in-line HAs offer lower latency than look-aside HAs because the former does not require any software mediation. However, in-line HAs tie the complete pipeline

of Figure 2.1 to the choice of a HA, thus limiting the advantages of virtualization in terms of resource sharing. Moreover, the performance gap between the two models is quickly closing [41], suggesting that look-aside HAs may become predominant.

2.3. GPU-based HA resource multiplexing

As stated in Chapter 1, GPU-based HAs are gaining the attention of mobile operators thanks to their capability of reusing computing resources for Artificial Intelligence (AI) and Machine Learning (ML) workloads [18]. For instance, lately NTT Docomo has rolled out GPU-accelerated 5G trials, and SoftBank has explored a proof-of-concept about multiplexing the resources of a GPU between 5G DU workloads and ML-based edge computing applications [17]. Motivated by the impact that AI can have to transform the Radio Access Network (RAN) ecosystem for 5G and 6G, important market players have recently founded the AI-RAN Alliance [42]. Also, NVIDIA, Ericsson, Nokia and TMobile have just created a partnership to study GPU-accelerated vRANs [43].

There are essentially three methods to multiplex GPU resources. One is application-level sharing, which leverages conventional single-process concurrency methods, such as Compute Unified Device Architecture (CUDA) streams and multi-threading. While remarkably simple, time-sliced context switching can create significant and non-controllable overhead, ruling out per-process performance guarantees that are critical in PHY-layer processing pipelines discussed in Section 2.1.

Process-level multiplexing can be achieved with Multi-Process Sharing (MPS), first introduced on NVIDIA's Kepler architectures and further enhanced on Volta. MPS assigns subsets of GPU compute units, known as Streaming Multiprocessors (SM), to individual partitions for specific processes. Although MPS offers SM isolation to circumvent context switching overheads, cache and memory resources are still shared between processes without any isolation, as illustrated in Figure 2.2. This can potentially lead to large overheads when memory-intensive processes compete for GPU resources.

Finally, hardware-level multiplexing can be achieved with Multi-Instance GPU (MIG), a feature appeared with Ampere GPUs (e.g., NVIDIA A100). As shown in Figure 2.2, MIG facilitates true hardware isolation, guaranteeing Quality of Service (QoS) and resource allocation. Specifically, MIG enables the division of a GPU into fully isolated sub-GPUs, which in practice only share the Peripheral Component Interconnect (PCI) interface bandwidth towards the Central Processing Units (CPU). While it ensures full isolation, MIG presents two notable drawbacks compared to MPS. First, MIG offers only a limited set of partition options — for example, the smallest partition in a 40GB NVIDIA A100 provides 1/7th of SMs and 5GB of memory. Conversely, MPS allows for more granular SM splitting. Second, MIG incurs substantial overhead, on the order of several seconds, as opposed to a few hundred milliseconds with MPS (see Chapter 5).

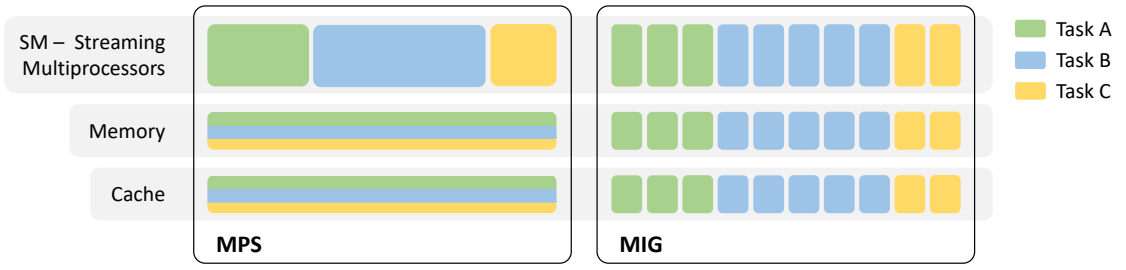


Figure 2.2: MPS versus MIG.

2.4. O-RAN

The O-RAN Alliance is a major carrier-led effort to define an open RAN architecture [7], depicted in Figure 2.3. Founded by AT&T, Orange, DT, Docomo, and China Mobile, over 300 industry players contribute to O-RAN today, including most major operators and vendors. The Control Plane (CP) of this architecture includes two components, the Non-Real-Time (Non-RT) RAN Intelligent Controller (RIC) (Non-RT RIC) and the Near-Real-Time (Near-RT) RIC, that use A1 and E2 interfaces to manage Network Functions (NF) such as DUs at, respectively, >1 s and >100 ms timescales. The User Plane (UP) has the O-Cloud, which provides computing resources, including CPUs and HAs, to NFs through an Acceleration Abstraction Layer (AAL) [44]. The AAL abstracts O-Cloud resources as *Logical Processing Units (LPU)*.

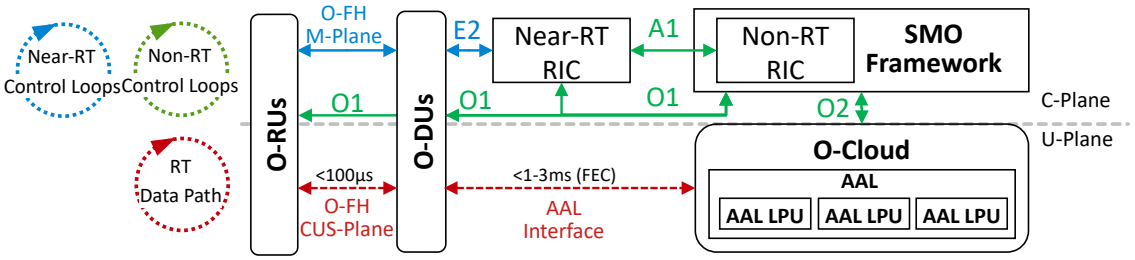


Figure 2.3: O-RAN architecture.

As shown in Figure 2.4, each LPU is dedicated to one NF via individual First In First Out (FIFO) queues. Consequently, though a physical processor (CPU or HA) can be shared among NFs, the state of each LPU (e.g., its queue occupancy) is *not* shared and thus only visible to the associated NF.

The O-Cloud is governed by the Service & Management Orchestrator (SMO) through the O2 interface, but operates on several-second timescales. Moreover, the Near-RT RIC lacks O-Cloud visibility, hindering compute-aware radio policies and DU coordination mechanisms at fast timescales — both crucial for achieving efficiency gains reliably.

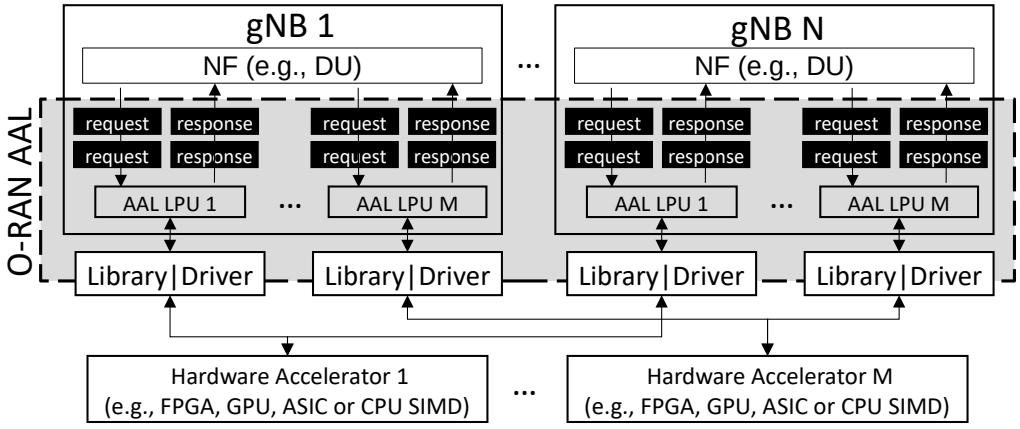


Figure 2.4: O-Cloud high-level architecture.

2.5. Related work

In this section, existing works on O-RAN control and virtualization, energy efficiency and multi-agent learning in mobile networks, task scheduling and HA resource multiplexing are examined, highlighting the reasons why these works are not suitable for answering the main research challenge of this thesis: the deployment of energy- and cost-efficient yet reliable hardware-accelerated vRANs (see Section 1.2).

2.5.1. O-RAN control and virtualization

The advent of O-RAN has recently motivated substantial research on radio control, management and orchestration. OrchestRAN [45] is a orchestration framework that determines the optimal set of data-driven models running in the O-RAN Near-RT RIC to offer intent-based control operations for sets of Base Station (BS) for mobile operators. Such an orchestration tool executes in the Non-Real-Time RIC and has been successfully validated through ColO-RAN [46], a comprehensive large-scale testbed equipped with software-defined radios-in-the-loop that evaluates the data-driven algorithms running in Near-RT. However, these orchestration solutions operate at second- or -minute-timescales and are not capable of cope with real-world workloads that fluctuate at smaller timescales, complicating the attainment of industrial hard reliability constraints.

The authors of [21] present an O-RAN control algorithm that employs Bayesian Learning theory to derive energy-efficient radio policies in vRANs while trying to maximize network performance. This work was subsequently extended in [47] to account also for co-located edge services. However, the system is controlled over long timescales and the computing resources are statically pre-assigned to both vRAN BSs and the edge services, hindering respectively strict reliability constraints and cost-effective deployment.

In Nuberu [13], a novel 5G DU is designed and engineered to ensure reliability on

non-deterministic and shared virtualized computing platforms by balancing reliability with network delay. While the use of reliable DUs is key in industry-grade vRANs, Nuberu neither coordinates multiple concurrent DUs nor relies on efficient mechanisms to optimally share the available computing resources, failing in providing an affordable cost per DU and thus a cost-efficient vRAN deployment.

In [22], the authors present a ML solution based on Deep Reinforcement Learning (DRL) and Deep Deterministic Policy Gradient (DDPG) that orchestrates CPU resources across multiple vRAN BS through control decisions taken depending on specific network contexts. Although this works faces the problem of sharing computing processors among many BSs to amortize costs and exploits compute-aware control decisions, the vRAN system is optimized only in long timescales and does not rely on any HA, which makes difficult meeting hard reliability constraints imposed by industry standards [48, 49].

Agora [30] is a 5G data channel processor built on top of FlexRAN's publicly available libraries for many-core general purpose CPU platforms that provides deterministic computing latency for massive Multiple-Input Multiple-Output (MIMO) baseband processing. The systems does not require the usage of HAs like ASICs, FPGAs or GPUs to deal with the baseband processing, and rather relies on the parallelism offered by dedicated CPU cores to meet the latency requirements of 5G NR. However, this solution is based on assigning dedicated resources and thus is not suitable for sharing available computing platforms to amortize deployment costs.

Similar to [30], Hydra [31] leverages Single Instruction Multiple Data (SIMD) programming capability of CPUs to scalably process in software massive MIMO workloads distributed across a pool of different servers. The system succeeds in supporting larger MIMO configurations than state-of-the-art approaches by yielding zero overhead when splitting the Fronthaul Network (FH) traffic from the RUs and reducing both inter-server and inter-core communication and coordination overheads. However, this work employs a distributed solution with multiple servers and does not apply DU centralization, thus hindering the cost-efficient deployment of vRANs.

In the context of resource multiplexing in vRANs, Concordia [12] employs a CPU scheduler based on a quantile decision tree to perform CPU resource multiplexing between co-located and co-executed 5G processing tasks of a single base station and other latency-elastic third-party applications. While Concordia meets the 5-nines reliability requirements on emulated 5G traces, it does not provide support to share heterogeneous pools of industry-standard HAs and CPUs among multiple DUs, thus presenting limitations in terms of cost-efficient vRAN deployment.

In [50], the authors present Atlas, a solution that minimizes connectivity disruption in commercial-grade 5G vRANs by sharing in Real-Time (RT) a common RU between two different DUs. With this system, the DUs implemented in software are made upgradeable and fault-tolerant, thus ensuring service continuity and resilience. While this work

represents a concrete step towards the support of resilient vRANs, it does not pool the available computing resources across multiple DUs to save deployment costs.

The authors of [51] propose EdgeRIC, an RT-RIC that provides DU control at sub-millisecond timescale through AI-based algorithms. Although EdgeRIC shows that RT solutions can achieve better system throughput than Near-RT solutions, it only considers scenarios with a single DU. Therefore, the RT control of multiple DUs and their access to a shared pool of computing resources including HAs, which are fundamental to reliably achieve higher efficiency gains and amortize installation costs, are not addressed.

ORANUS [52] is an O-RAN compliant system that dynamically assigns radio resources in both Near-RT and RT to ensure performance guarantees for multiple Ultra-Reliable Low-Latency Communication (URLLC) services. These services are mapped to multiple DUs deployed in the same cell and are assigned with RBs through a RT transmission-queue-aware radio allocation policy. However, this work focuses only on a single-cell scenario and does not consider dense industry-grade scenarios with several cells, in which HAs are needed to guarantee typical reliability targets for processing vRAN functions.

In OREO [53], authors propose a control framework that operates in Near-RT to efficiently assign CPU resources to services running in the vRAN. To guarantee target requirements for these services, the solution takes into account complex trade-offs in the dimension of latency, quality of service and resource demands. However, this framework operates at coarse time granularity and thus is not suited to handle dynamic real-world workloads, making it challenging to meet the strict reliability targets set by industry.

2.5.2. Energy efficiency in mobile networks

Energy efficiency in mobile networks has been traditionally researched with the primary focus on the energy consumed by the BSs for amplifying wireless signals (a.k.a. transmission power). Numerous studies and literature address this issue, delving into analytical models and deployment planning tools derived from these models, as in [54–56].

The current trend in network densification involves deploying a higher number of small base stations with lower transmission power and higher data rates [57]. Due to network virtualization, the computing capacity needed to process high-bitrate signals is now a significant factor in the RAN energy consumption, as observed in [58, 59].

A few studies focus on vRANs energy consumption: for instance, the works [22] and [47], already presented in Subsection 2.5.1, respectively use CPU resources and energy-efficient radio policies to reduce the energy footprint of the vRAN. However, none of these consider hardware acceleration, crucial for industry-grade systems [48, 49].

2.5.3. Multi-agent learning in mobile networks

Previous Multi-Agent Reinforcement Learning (MARL) methods often handle only a small number of agents due to the exponential increase in complexity with the number of agents, known as *the curse of dimensionality* [60]. A few works tackle the scalability issue by using concepts from Mean Field Theory [61, 62]. In these studies, interactions within the agent population are approximated by those between a single agent and the population's average effect. While these techniques were applied in fields like the control of Unmanned Aerial Vehicles (UAV) [63] and the management of ride-sharing platforms [62, 64], they were never applied to mobile networks to the best of our knowledge.

In mobile networking, single-agent Reinforcement Learning (RL) finds extensive use in spectrum management [65], network diagnostics [66], software-defined networking [67], among others. In mobile network problems, the Markov Decision Process (MDP) is often particularized with a *contextual bandit* due to several reasons. First, these problems typically have an infinite horizon, with the goal of optimizing long-term performance, leading to a zero discount factor. Second, state transitions are often independent of the selected action. Finally, the reward observation is usually not delayed. Thus, numerous works express mobile network challenges through contextual bandits [22, 47, 59, 68–70].

Despite the potential benefits, the implementation of an arbitrarily large number of agents (DUs) collaboratively solving a contextual bandit problem has not yet been explored in the context of mobile networks. This gap in research hinders the development of an economically effective strategy that relies on cost amortization for hardware-accelerated DUs through the pooling of HAs across multiple centralized DUs.

2.5.4. Task scheduling

Considering task scheduling frameworks, [71] propose a Mixed-Criticality Federated Scheduling (MCFS) algorithm to assign computational resources to safety-critical and non-safety critical tasks that are sharing a common computational platform in order to meet different processing deadlines. The solution responds to increasing computational demands of inelastic tasks by providing internal parallelism via assignation of cores and virtual deadlines. In [72], authors present ZYGOS, a solution to efficiently schedule networking tasks on multi-core computing platforms with the objective of reaching high throughput, expressed as number of tasks timely processed, while guaranteeing μs-level tail latency. Arachne [73] is a core-aware system that assigns applications of few microseconds duration to dedicated cores for processing, with the number of cores assigned depending on the specific load of the application. Arachne goal is to reduce tail latency and provide high throughput in terms of timely processed applications. However, these works [71–73] operate at coarse scheduling time granularity, which is not suitable for vRAN workloads that are fluctuating at sub-millisecond timescales.

Shinjuku [74] improves [72] by implementing a centralized scheduler that uses hardware-supported virtualization to perform tasks preemption at few microseconds timescale. This solution effectively copes with real-world tasks processing times showing distributions with both light and heavy tail, providing consistent gains in terms of throughput and tail latency. Shenango [75] focuses on achieving microsecond-scale latencies for datacenter workloads with an efficient usage of the available CPU resources that limits overprovisioning. This is obtained by reallocating CPU unused cores at a granularity of 5μs from latency-sensitive tasks, for which an overprovisioned amount of cores has been allocated to deal with the peak expected load, to batch processing tasks. Snap [76] is a userspace system for the implementation and the execution of several network functions that are assigned with dynamically scaled CPU resources through a scheduler operating in real-time. While [74–76] can provide μs-level scheduling and reallocation strategies, they are generic approaches alien to the specifics of radio scheduling operations, hence are not suitable for sharing heterogeneous computing platforms while providing many-9s reliability in dense vRAN scenarios.

2.5.5. GPU resource multiplexing

The objective of this subsection is to shed light on existing works focusing on the multiplexing of GPU computing resources. Indeed, this approach can help reducing deployment costs compared to more expensive solutions in which individual GPU-based workloads are assigned with dedicated GPUs.

Motivated by the fact that SM cores of entire GPUs are often underutilized, [77] provides an exhaustive characterization in terms of performance and energy consumption of an A100 GPU in different MIG mode operations with various state-of-the-art Deep Learning (DL) models. This work demonstrates that MIG sub-portions of the GPU can support these DL workloads, thus avoiding hardware resource wastage resulting from using the entire GPU. However, as demonstrated next in this thesis (see Chapter 5), MIG struggles with the dynamic multiplexing of GPU resources due to high reconfiguration overheads, which cast doubts about the usage of this mechanisms when dealing with GPU resource reconfiguration for dynamically changing DU and ML co-located workloads.

MISO [78] is a technique that uses MPS-based resource multiplexing to predict and subsequently dynamically allocate the most effective MIG partitions for diverse tasks running on GPU, thereby mitigating the challenge and the overhead of exploring different partition configurations with MIG alone. Yet, MISO fails to address the issue of reconfiguring slices of the GPU for dynamically changing workloads, such as RAN traffic.

Ligthweight GPU multiplexing is offered by Salus [79] and TGS [80]. Salus enables two GPU multiplexing primitives at low overhead: quick job switching and memory sharing, fostering fine-grained GPU multiplexing many different GPU-based applications by means of iteration scheduling and resolution of memory management issues. TGS provides

transparent GPU multiplexing among DL training workloads in container clouds by operating at the Operating System (OS) layer rather than the application-layer. Through adaptive rate control and transparent unified memory, TGS ensures high GPU utilization and performance isolation, i.e. opportunistic tasks do not affect co-located production tasks. Nevertheless, these solutions do not adequately handle the two fundamentally disparate workloads generated by DUs (inelastic) and by ML tasks (elastic), thus are not applicable in the context of cost-efficient deployment of GPU-accelerated vRANs leveraging ML algorithms to improve DUs operation.

3 Experimental analysis of vRAN acceleration for FEC decoding

As discussed in Chapter 1, the sustainable deployment of hardware-accelerated virtualized Radio Access Network (vRAN) that meets strict reliability constraints while reducing deployment costs and energy consumption has become an important and challenging problem to solve for network operators. Indeed, current industry-grade vRANs rely on expensive and energy-hungry Distributed Unit (DU)-dedicated Hardware Accelerators (HA) to ensure the 5-nines processing latency probability and tend to shun other slower yet more energy-efficient processors like Central Processing Units (CPU).

This thesis demonstrates that it is possible to deploy energy- and cost-efficient yet reliable vRANs respectively by (*i*) opportunistically complementing HAs with less energy-hungry computing processors like CPUs to process DU tasks, and (*ii*) sharing the same HA across multiple centralized DUs as to amortize the cost of the expensive HAs.

This chapter presents a set of experimental results that form the foundation for and justify the aforementioned design choices needed to deploy energy- and cost-effective yet reliable vRANs. In Section 3.1, typical real-world DU workloads used throughout this thesis are presented to prove that the proposed solutions are validated with real Radio Access Network (RAN) traces. In line with the discussion in Section 1.1 exemplified in Figure 1.3, Section 3.2 provides insight about the reasons why HAs are preferred over CPUs in industrial vRANs for 5G Forward Error Correction (FEC) decoding under real-world traffic. Section 3.3 shows that CPUs can be valuably complemented with HAs in vRANs to provide advantages in terms of capital and operating costs, thus justifying the *opportunistic HA offloading model*. In Section 3.4, typical utilization figures of HA dedicated to individual DUs deliver practical support for cost-efficient exploitation of HA resources by means of *sharing single HA among multiple centralized DUs*. Equivalently, these results validate the opportunity of multiplexing spare resources of the HA with other co-located services: in Chapter 5, we will show a realistic demonstration of this approach, considering the specific example of Graphics Processing Units (GPU)-based HAs.

3.1. Real-world DU workloads

Using Falcon [81] and 5GSniffer [82], we tracked the workload dynamics experienced by several (sub)urban cells in Madrid, Spain (Vodafone, April 2021, ESP), in Frankfurt, Germany (Deutsche Telekom, December 2022, DE), and in Boston, US (T-Mobile, May 2023). The goal is to use this data to emulate the behavior of real User Equipment (UE).

Consistent with previous studies [12,83], we observe that the individual loads and the number of concurrently active UEs are low at the Transmission Time Interval (TTI) scale. The "x1" curves in Figure 3.1, which characterize each UE's buffer (left), inter-arrival time (middle), and Signal-to-Noise Ratio (SNR) (right) at the TTI level, show a median and 99th percentile UE buffer of 2 Kb and 78 Kb, respectively. Moreover, Figure 3.2 indicates that the median and 99th percentile of active UEs in one TTI is only 1 and 6, respectively.

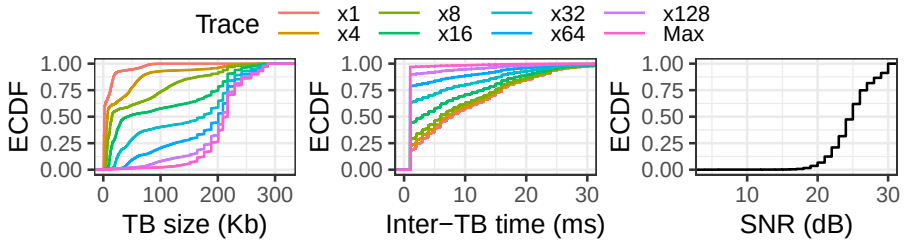


Figure 3.1: UE demand profiles used for evaluation.

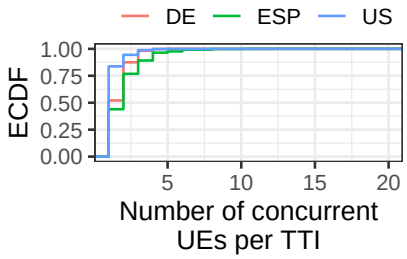


Figure 3.2: UEs concurrently active at TTI timescales.

To emulate higher network demands, we generated additional profiles by amplifying our traces by factors of 4, 8, and so forth, denoted as "x4", "x8", etc. in Figure 3.1. With the mobile traffic Compound Annual Growth Rate (CAGR) pegged between 25-30% [84], these multipliers enable us to project expected workloads up to 2030, all the while preserving the genuine dynamics of real users. The additional "Max" profile, where the UE is consistently backlogged, let us analyze worst-case scenarios too.

3.2. Reliability of legacy CPUs and HAs

The industry today favors in-line or look-aside HAs such as Application-Specific Integrated Circuits (ASIC), Field Programmable Gate Arrays (FPGA), or GPUs for 5G FEC workloads due to concerns about CPU reliability [85]. Indeed, there exist fundamental limitations to parallelizing individual decoding request across CPU cores [86], hence state-of-the-art Low-Density Parity-Check (LDPC) libraries such as FlexRAN [87] and others [88,89] just exploit data parallelization using Single Instruction Multiple Data (SIMD) programming and task parallelization over concurrent decoding tasks [30, 31]. Yet, those strategies cannot bound the latency of *individual* tasks, which prevents CPUs from guaranteeing reliability for *all* workloads alone and limits the applicability of these processors in industry-grade vRAN scenarios.

To precisely quantify the reliability that CPU- and HA-based processors may achieve, we measure in Figure. 3.3 the ratio of Transport Block (TB) processed within a deadline D (reliability) for two solutions. The first one uses FlexRAN, a de-facto standard library [12, 30, 31], with the above optimizations on a pool of Intel Xeon Gold 6240R CPU cores. The second one uses a commercial NVIDIA V100 GPU decoder, which represents a new and still little understood HA with compelling features like Compute Unified Device Architecture (CUDA), CPU-like time-to-market, and AI-on-5G [36].

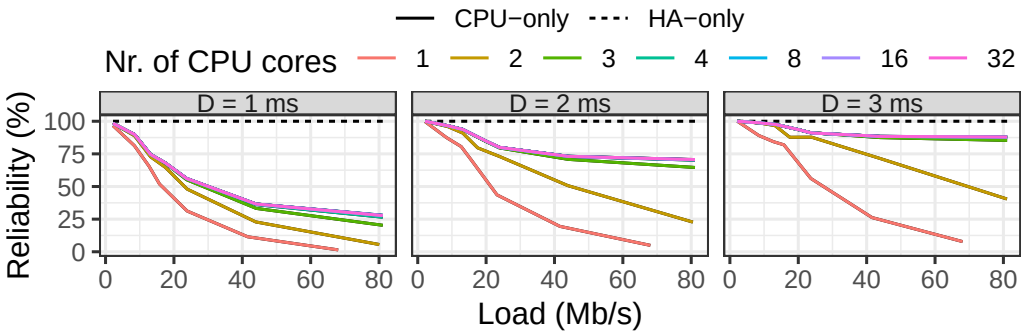


Figure 3.3: Reliability performance as a function of the DU load for a CPU-only processor and a HA-only processor.

We evaluate both processors for different loads by varying the profile of the active UEs as explained in Section 3.1 (“x1”, “x2”, “x4” and so on). Every TTI, a number of UEs become active following the distribution shown in Figure 3.2, each receiving a fair amount of the radio resources from a 100-MHz DU. Then, these UEs generate TBs following the corresponding profile and radio allocation by executing the downlink pipeline of Figure 2.1. To emulate the wireless channel, we add Additive White Gaussian Noise (AWGN) noise with zero mean and the appropriate variance to obtain the target SNR across all the Orthogonal Frequency-Division Multiplexing (OFDM) symbols received by the DU.

To ground our tests on realistic settings, we implemented O-RAN's Acceleration Abstraction Layer (AAL) using Data Plane Development Kit (DPDK)'s Wireless Baseband Device Library (BBDev) [90], which provides abstractions for wireless processing tasks that can be used to implement the Logical Processing Units (LPU) for both processors [91] (∼1K lines of C code). The signals demodulated by the DU are encoded as soft bits, as shown in the uplink pipeline of Figure 2.1, stored in a memory pool (*mempool*), and then allocated to the corresponding LPU.

As shown in Figure 3.3, the GPU (dotted black line), just like any look-aside or inline HA, can sustain 100% reliability regardless the load. Conversely, the CPU-only approach can only meet the 99.999% reliability target for small loads. This is naturally worse for smaller CPU pools as queuing effects cause additional delays but reliability drops regardless the pool size because higher loads carry larger TBs—which can exceed deadlines on CPUs—with higher probability.

These results illustrate the underlying reasons why CPUs are ignored by the industry for 5G FEC. *In contrast, we show next that CPUs can be a valuable complement to HAs.*

3.3. Complementarity of CPUs and HAs

We next experimentally investigate the performance of both categories of 5G FEC processors, unveiling their complementarities. For the CPU solution, we set the pool to 16 cores which our earlier results proved to be sufficient (see Figure 3.3). For each test, we select a Modulation and Coding Scheme (MCS) $m \in \mathcal{M} := \{0, 1, \ldots, 27\}$ (see [92, Table 5.1.3.1-2]), an SNR $s \in \mathcal{S} := \{1, 2, \ldots, 30\}$ dB, and a bandwidth $b \in \mathcal{B} := \{1, 2, \ldots, 250\}$ Resource Block (RB) to generate 10 random TBs, adding up to a total of ∼2M samples.

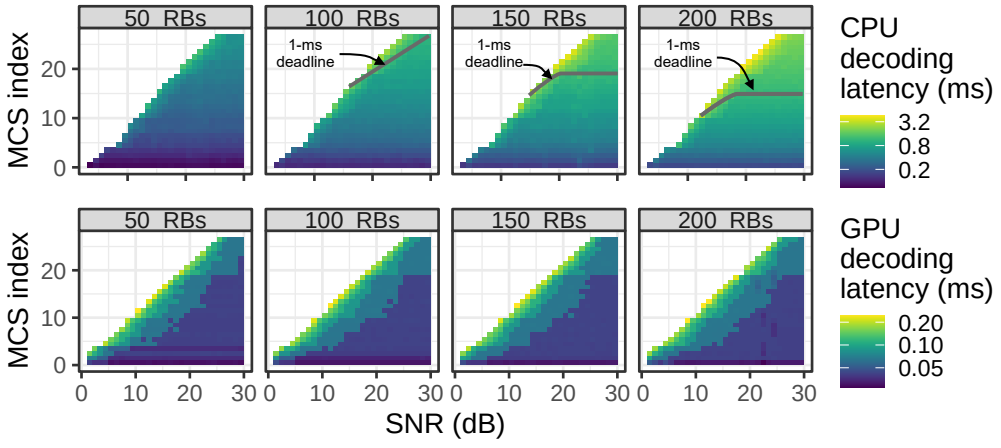


Figure 3.4: Median latency in decoding TBs of different sizes (i.e., combinations of RBs and MCS) under varied SNR conditions with CPU and HA processors. Log-scale z-axis.

3.3.1. Latency

Using a colored gradient, Figure 3.4 shows the median latency of the CPU running FlexRAN (top) and of the GPU-based HA (bottom) for all combinations from $\mathcal{S}$ and $\mathcal{M}$ and for subset of $b \in \{50, 100, 150, 200\} \subset \mathcal{B}$. The blank cells indicate contexts that cannot be decoded within 10 iterations. The HA provides roughly an order of magnitude improvement in latency, as expected. More interestingly, Figure 3.4 unveils for the first time key properties of the latency achieved by a GPU-based 5G HA, such as its invariance to the TB size or complex relationship with MCS and SNR combinations.

The latency performance of a 5G FEC processor depends not only on the type of processor but also on the decoding algorithm and library-specific implementations. Figure 3.5 compares FlexRAN and two open-source libraries, "OS1" [89] and "OS2" [88], depicting substantial differences across them. First, both FlexRAN and OS2 provide, on average, 4.3 times lower latency than OS1. This is explained because our OS1 library under test does not exploit SIMD acceleration. Second, OS2 is unable to decode some high-MCS, high-SNR TBs (see blank cells in the plot). The reason lies in the 8-bit quantization used in this specific library to encode soft bits. We repeated the experiments with 16-bit resolution and the processor could decode those TBs as expected but with a much larger latency toll (right plot). Finally, there are mild—yet relevant—differences in performance when comparing *decodable* TBs with FlexRAN and OS2. More specifically, FlexRAN outperforms OS2 by around 10% in average, but by over 30% when considering TBs that require more than one iteration to decode, even though they both exploit SIMD. The reason is that the three libraries each implement a different flavor of LDPC decoding, which renders a different number of decoding iterations under the same noise.

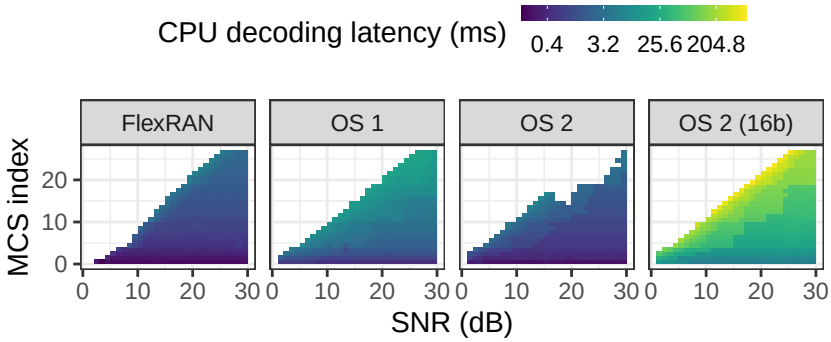


Figure 3.5: Median latency in decoding TBs of different size, with 200 RBs and varied SNR conditions, using different CPU libraries including FlexRAN, srsLTE and OpenAirInterface. Log-scale z-axis.

These results prove that also CPU solutions can decode TBs within common 1-3 ms deadlines [12, 13] in a wide range of contexts (those below the grey line in the Figure 3.4).

In fact, 100% of the TBs observed in the "x1" UE profile fall within this range. Moreover, the results show that there exists a complex relationship between TB context (MCS, SNR, number of RBs), the type of the processor, the decoder implementation, the amount of computing resources assigned, and performance, which suggests data-driven models and challenges the formulation of tractable mathematical models.

3.3.2. Energy consumption

Figure 3.6 shows the energy consumed by the CPU and the GPU processors, considering respectively the FlexRAN implementation and 100% of allocated Streaming Multiprocessors (SM)s. For the less demanding bandwidth setting, the HA consumes more than 28× the energy required by the CPU. Yet, for the most exacting bandwidth configuration, the HA consumes *only* around 71% more than the CPU. The latency provided by CPUs precludes their use in specific contexts; however, the remarkable advantages in terms of capital (see Table 1.1) and operating (i.e., energy) costs still make CPUs appealing to process less-stringent TBs, which are rather frequent as discussed before.

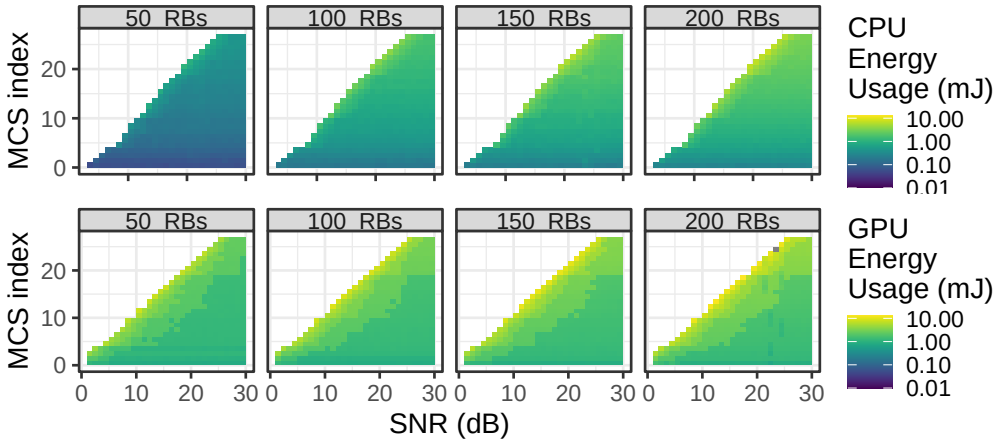


Figure 3.6: Median energy to decode TBs of different size (i.e., combinations of RBs and MCS) under varied SNR conditions with CPU and HA processors. Log-scale z-axis.

3.4. HA sharing and multiplexing opportunities

Let us now explore HA sharing and multiplexing opportunities as a means to improve cost-efficiency. Figure 3.7 depicts the relative busy time of the GPU-based HA (y-axis) to process the workload of a varying number of concurrently active UEs (x-axis) with different profiles (colors) in a single 100-MHz 5G DU. For "x1"-generated workloads, the HA utilization is below 12% even with 10 concurrently active UEs.

In fact, 10 “Max” UEs concurrently active every TTI—or 85 “x4” UEs, though not shown in the figure—would be required to saturate the HA. In conclusion, modern HAs are largely underutilized by individual DUs when handling real-world workloads, which creates clear opportunities for sharing and multiplexing, even in a relatively far future.

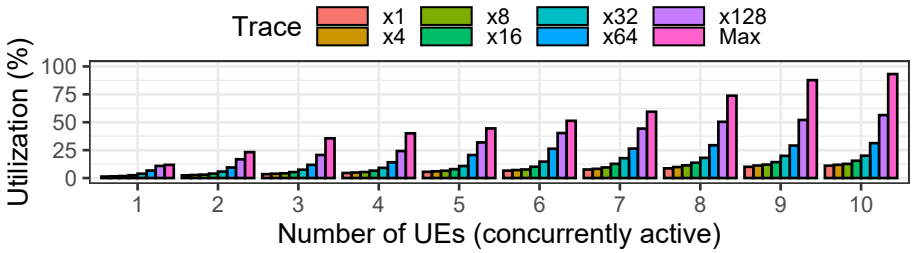


Figure 3.7: GPU HA mean usage for different 5G workloads.

A legitimate ensuing question is whether HA sharing among multiple DUs can be sustained by the Fronthaul Network (FH) [93] that connects them to their corresponding Radio Unit (RU) as shown in Figure 2.3. O-RAN uses a 7-2x DU-RU split with an open FH interface based on Evolved Common Public Radio Interface (eCPRI) that has a latency tolerance of 100 μs [94]. Assuming a 5-μs/Km propagation delay and a 2D Manhattan tessellation model, the area of RUs that can be connected to a single location is up to 400 Km^2 [37]. Moreover, assuming Peripheral Component Interconnect Express (PCIe) v3.0+ bus and 100-GbE FH interfaces, we could aggregate up to 3.8 GHz of radio spectrum per server [95]. Hence, it is technically feasible to aggregate the workload of multiple DUs and share the resources of high-performing HAs in centralized edge clouds.

The underutilization of DU-dedicated HA with real-world workloads also opens to the unique opportunity of re-using spare HA resources to handle tasks external to the DU Physical layer (PHY) processing to build more cost-effective vRAN systems. As already discussed in Section 1.1, GPUs have been proposed as a realistic candidate for HA thanks to their unique capability of handling GPU-based Machine Learning (ML) models that can be used to improve DU operations. Yet, multiplexing the available resources of the GPU poses novel challenges and trade-offs, that will be analyzed in the Chapter 5.

4 Energy and cost efficiency in vRANs with Near-RT control

The analysis performed in Chapter 3 sets a roadmap for more energy- and cost-efficient virtualized Radio Access Network (vRAN) deployments. In particular, the results in Section 3.3 have shown that Central Processing Units (CPU) can complement energy-hungry Hardware Accelerators (HA) like Graphics Processing Units (GPU) to process some of the compute-intensive tasks running at the Distributed Unit (DU) of virtual Base Station (BS), trading off energy savings with processing latency gains. The findings of Section 3.4, instead, have demonstrated that HA can be shared among multiple centralized BSs as to improve cost efficiency by amortizing the acceleration cost of individual BSs.

This chapter presents the first solution of the thesis that aims at deploying energy- and cost-effective vRANs by leveraging the aforementioned two design approaches. According to the analysis of Section 3.3, there exists complex relationships among the network context, the type of processor and the latency and energy performance. For this reason, the proposed solution harnesses data-driven policies to control the offloading strategies of multiple BSs that are centralized in the same shared computing platform.

4.1. Offloading scenario in vRAN

The novel O-RAN architecture defines a computing platform known as O-Cloud to offload signal processing workloads from virtualized BS. Among these workloads, we focus on the Forward Error Correction (FEC), which is the most compute-intensive [13]. An O-Cloud provides signal processors comprised of general-purpose CPUs and HAs such as FPGAs, GPUs, or ASICs. Each processor queues FEC processing requests in FIFO queues and, once processed, the resulting Transport Block (TB) data is sent back to the associated BS (see Figure 4.1). In O-RAN, BSs are controlled by a Near-Real-Time (Near-RT) RAN Intelligent Controller (RIC) using apps, known as *xApps*, which operate in the timescale of $\sim 10 - 100$ ms (i.e., 10x or 100x longer than a Transmission Time Interval (TTI)). In this chapter, we focus on a data-driven policy in a Near-RT-RIC to control the offloading strategy of BSs deployed over a prototype O-Cloud platform.

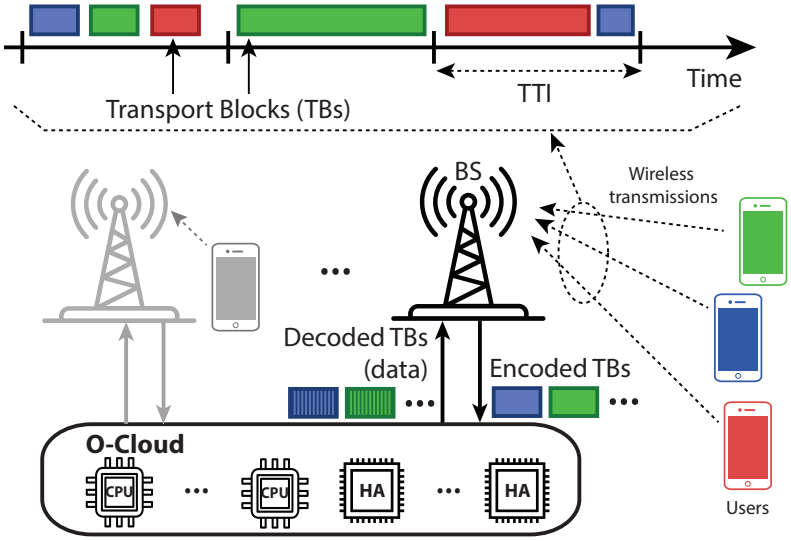


Figure 4.1: O-Cloud offloading scenario in vRAN.

4.2. Problem formulation

In this section, we formulate an opportunistic HA offloading problem for 5G signal processing workload as a discrete-time decision-making problem where each time step k corresponds to a TTI. We let $\mathcal{B}_k$ denote the set of B_k BSs that share the same O-Cloud platform at time step k. Note that the number of active BSs may change over time. Every time step k, each BS $b \in \mathcal{B}_k$ receives from its users a set of encoded TBs $\mathcal{T}_k^b$ that must be FEC-decoded by the O-Cloud platform. Every TB $d_i \in \mathcal{T}_k^b$ is characterized by its Signal-to-Noise Ratio (SNR) (c_i), Modulation and Coding Scheme (MCS) (m_i), and the amount of data bits it carries (l_i). BSs must assign each encoded TB to one type of processor: a software processor (no offloading) or a hardware accelerator (offloading). We let $\mathcal{A}_k^b$ denote the set of assignment actions taken by the BS $b \in \mathcal{B}_k$ at time k. Each individual action $\alpha_i \in \mathcal{A}_k^b$ is associated with a TB $d_i \in \mathcal{T}_k^b$. Therefore, both sets have the same cardinality ($|\mathcal{A}_k^b| = |\mathcal{T}_k^b|$), which may change over time steps k, i.e., the number of TBs generated by a BS can change for different TTIs. Evidently, the individual action $\alpha_i \in \mathcal{A}_k^b$ takes binary values. When $\alpha_i = 0$, $d_i \in \mathcal{T}_k^b$ is assigned to a CPU queue (software processing), and when $\alpha_i = 1$, the TB is assigned to the HA queue (for hardware acceleration). We also let $\mathcal{A}_k$ denote the joint set of actions, i.e., $\mathcal{A}_k := \{\mathcal{A}_k^b \mid b \in \mathcal{B}_k\}$.

We now let $\gamma(\alpha_i, d_i)$ and $q(\alpha_i, d_i)$ denote the processing and waiting/queuing time of d_i, respectively. Note that $\gamma(\cdot)$ and $q(\cdot)$ not only depend on the selected resource for processing (software or HA) as we show in Figure 1.3, but also on the features associated with the TB , i.e., c_i, m_i, and l_i. In line with the industry standards, we assume that all the CPU cores are identical as well as all the HAs in the O-Cloud.

According to the specifications, every TB in the system older than a time deadline τ is discarded, incurring data loss. A TB can be discarded while waiting in a queue ($q(\alpha_i, d_i) > \tau$) or after being processed ($\gamma(\alpha_i, d_i) + q(\alpha_i, d_i) > \tau$), whatever happens first. We assume a long enough queue in the HA pool to avoid TB losses due to queue overflow.

Note that, although $\gamma(\cdot)$ and $q(\cdot)$ are continue-valued, i.e., a processing task can finish between two consecutive time steps, the decision intervals (every TTI) are discrete, and hence the problem can be formulated in discrete time. However, as processing a task can take more than one time step, the actions taken in k may affect the system state in future time steps. We now let $E_k := \sum_{b \in \mathcal{B}_k} \sum_{d_i \in \mathcal{T}_k^b} e(\alpha_i, d_i)$ denote the energy consumed by the O-Cloud due to the transport blocks generated at time step k, where $e(\alpha_i, d_i)$ corresponds to the energy consumed by the processor selected by action α_i to process TB d_i. Similarly, we define the ratio of TBs successfully processed in the O-Cloud within the deadline during the time step k as $\zeta_k \in [0, 1]$.

We then formulate our opportunistic HA offloading problem as follows:

Problem 1.

$$\begin{aligned} \min_{\{\mathcal{A}_k\}_{k=1}^K} \quad & \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} E_k \\ \text{s.t.} \quad & \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} \zeta_k \geq 1 - \epsilon. \end{aligned}$$

where ϵ sets the target reliability (e.g., $\epsilon = 10^{-5}$ in [12]).

Problem 1 assumes that the O-Cloud system is dimensioned with sufficient computing resources (HAs and CPUs) to operate *always* within the problem's feasibility region, but this may not necessarily hold always. If that is the case, the goal is to maximize throughput regardless of energy consumption:

Problem 2.

$$\max_{\{\mathcal{A}_k\}_{k=1}^K} \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{T} \zeta_k$$

Solving these problems is highly intricate. Firstly, O-Cloud energy consumption relies on computing resource utilization (CPU and HAs), influenced by the number of generated TBs from the BSs, the selected actions $\mathcal{A}_k$, and TB characteristics (c_i, m_i, and l_i) at time step k and previous steps. Notably, a TB lost at time k might have been assigned in a prior time step (TTI $< \tau$), and queued TBs add up waiting time to TBs in the near future. Secondly, TB processing time and energy consumption involve a random component, as observed in our experiments in Section 3.3. This randomness pertains to how bits are ordered within a TB and how an FEC decoder extracts information

(see [40]). Thirdly, both TB energy consumption and processing time depend on O-Cloud hardware specifics and software processing implementation, which can vary across platforms or evolve over time due to hardware/software upgrades. Figure 4.2 illustrates processing time variations for different software and HA decoder implementations and configurations. Lastly, determining which problem to solve (Problem 1 or 2) is inherently challenging as it depends on unknown system dynamics.

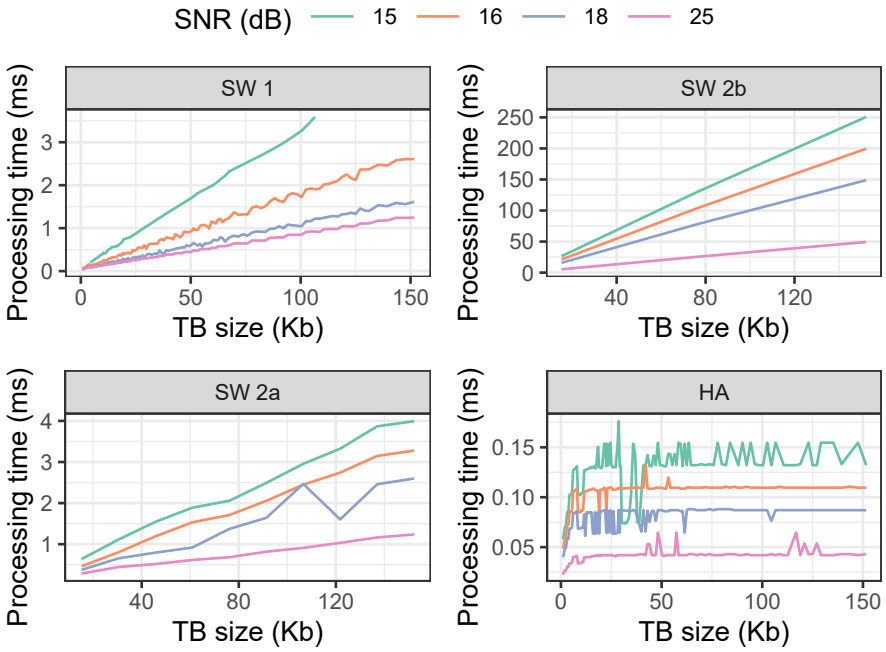


Figure 4.2: Processing time of a TB as a function of its size for different SNR values and MCS index 15. We evaluate a HA, two different implementations of the software decoder (SW 1 and 2), and two configurations of the second software decoder (SW 2a and 2b).

Therefore, our main problem cannot be solved analytically. One typical approach is to model it as a Markov Decision Process (MDP) and employ Deep Reinforcement Learning (DRL). However, this is not feasible due to time constraints. *xApps* make decisions every 100 ms, while computing resource allocations need to be made every TTI $\leq$ 1ms. Even with relaxed time constraints, the problem remains challenging due to a vast action space, involving $2^{|\mathcal{A}_k|}$ possible actions, where $|\mathcal{A}_k|$ can be arbitrarily large. Additionally, the action space dynamically changes over time, contrasting with the typical Reinforcement Learning (RL) assumption of a fixed action space. Furthermore, our primary goal is to minimize the system's energy cost. Using large models to solve our problems may induce a noticeable energy burden on the system. To avoid this, we must design a lightweight, practical, and scalable solution capable of real-time operation at BSs with negligible energy and economic impact.

4.3. An efficient offloading strategy

We propose a practical and efficient energy-aware offloading strategy for vRANs. We rely on the insights from our experimental campaign and we exploit the problem described in Section 4.2 to devise an efficient solution in terms of energy and cost.

Minimizing the use of HAs can bring important energy savings. However, an excessive use of CPUs may induce throughput loss as they are a magnitude slower than energy-consuming HAs (see Figure 1.3). To get additional insights, we implemented an O-RAN-compliant O-Cloud platform comprised of an NVIDIA GPU V100 as HA and up to 16 Intel CPU cores for signal processing; Figure 4.2 shows the processing time $\gamma(\alpha_i, d_i)$ of a TB d_i with different TB sizes and SNR values, for 3GPP MCS index 15 [92]. The "SW" plots show the latency incurred by different implementations of a software processor on the same CPU ($\alpha_i = 0$), and the "HA" plot that by the GPU-based HA ($\alpha_i = 1$).

We observe a key structural difference between CPU and HA processing. The processing time of a CPU is highly dependent on l_i (TB size), while such dependency practically vanishes in the case of the HA. This structure holds even across different software implementations. Similar behavior can be observed for energy consumption as it is proportional to relative processor busy time. Based on these observations, we propose an intuitive strategy by which small TBs are processed in CPU, to avoid an early saturation of this kind of processor. Conversely, as the processing time of the HA is not sensible to the bit size of the TB, large TBs are offloaded. To this end, we define l_{th} as the TB size threshold or *bit threshold* that delimits two operational regions. When $l_i < l_{\text{th}}$, d_i is processed in CPU, otherwise the HA is used.

In order to evaluate the impact of l_{th} on the performance of the network, we set up a scenario with up to 10 BSs sharing our O-Cloud experimental platform with 4 Intel CPU cores dedicated to software processing and the NVIDIA GPU V100 as HA. We assume (for now) that the workload generated by all the BSs is independent and identically distributed (i.i.d), which implies that the l_{th} is common to all the BSs and simplifies analysis. We will relax these assumptions later. In this way, each BS handles the traffic of 3 homogeneous users, which generate data following a Poisson process with mean 70 Mb/s, and we simulate a Rayleigh wireless channel model with a random mobility pattern with 20 Km/h velocity [96]. We assess two scenarios: a high-load scenario with 10 BSs, and a low-load scenario with 2 BSs. It is worth noting that these scenarios are cost-efficient since the BSs share the same O-Cloud platform and achieve respectively 10x and 2x cost gains with respect to scenarios in which individual platforms are dedicated to individual BSs. For each scenario, we evaluate 30 different values of l_{th} during 0.5s and for 10 independent runs. Figure 4.3 depicts the mean and the 10^{th} and 90^{th} percentiles of the system reliability ζ (left) and the power consumption (right) for both low-load (top) and high-load (bottom).

We observe in the top part of Figure 4.3 that, when $l_{\text{th}} < 75 \cdot 10^3$, the ratio of processed TBs (ζ) is maximum. This means that only the TBs with a smaller size than the selected l_{th} are processed using CPUs and the rest of them are offloaded into the HA. For higher values of the bit threshold $l_{\text{th}} > 75 \cdot 10^3$ the CPU gets saturated and some TBs are dropped ($\zeta < 1$). On the other hand, the more we use the CPU (higher values of l_{th}), the higher the energy savings. In that case, as the system is not saturated, we are solving Problem 1 and the optimal point (energy saving maximization without data losses) is around $l_{\text{th}} = 75 \cdot 10^3$. In the bottom part of Figure. 4.3 we can observe that the higher traffic saturates the O-Cloud, i.e., $\zeta < 1 \forall l_{\text{th}}$. In this case, we need to solve Problem 2, and find the configuration that maximizes ζ. In that case, the optimal configuration is around $l_{\text{th}} = 25 \cdot 10^3$. Note that due to the higher performance of this configuration, the energy consumption is also maximized.

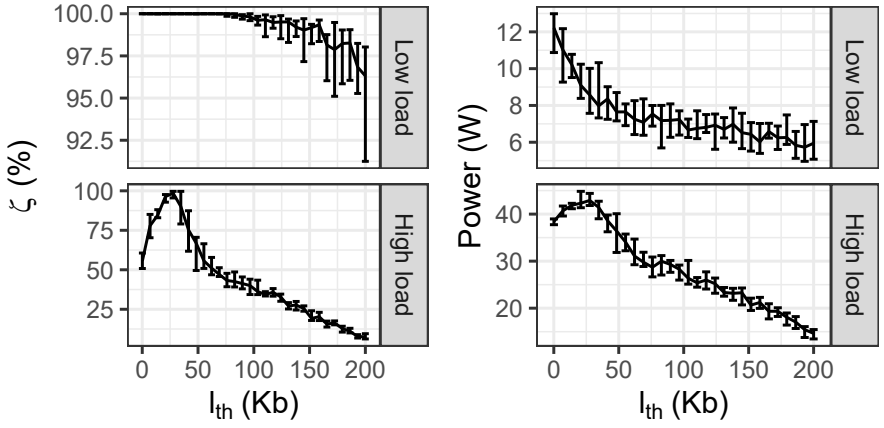


Figure 4.3: Ratio of successfully processed TBs (left) and power consumption (right) for low (top) and high (bottom) traffic loads.

From these experiments, we can extract an important conclusion. *The optimal value of l_{th} depends on the traffic conditions.* In consequence, we need to devise a strategy to find the optimal bit threshold of each BS depending on the system state in terms of traffic conditions. This strategy needs to be learned for each specific deployment, as it also depends on the hardware in the O-Cloud and the software implementation of the decoders (see Figure 4.2). Thus, the learned strategy can be deployed at the Near-RT-RIC, providing different configurations according to traffic variations.

Nevertheless, this problem is still very challenging to solve in a centralized way for several reasons. The O-Cloud workload is not only defined by the load of TBs (number of TBs per second), but also by the parameters c, m, and l of each TB, which have an impact on the processing time. Second, as the O-Cloud is common for all the BSs considered in the problem, we face a collaborative problem in which the decisions of each agent (BS

offloading configuration) affect all the others. For example, a poor action by one BS may saturate the CPU, increasing energy consumption or even causing throughput loss in some other BS sharing the O-Cloud. Third, the number of BSs can be arbitrarily high, especially in highly populated areas. Moreover, we consider the case where the BSs can be switched off and on dynamically, which is a popular feature in 5G [97]. This defines state and action spaces with changing dimensionality over time. For these reasons, we next formulate the problem as a Multi-Agent Contextual Bandit, and propose a practical algorithm using ideas from mean field theory.

4.4. Multi-agent formulation

4.4.1. Background: Markov games

A Markov game is defined by the tuple $\Gamma = \langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$, where $\mathcal{N}$ is the set of players or agents, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{R}$ is the set of reward functions, and $\mathcal{P}$ is the set of transition probability functions.

Let $\mathcal{N}_t \subseteq \mathcal{N}$ denote the set of N_t agents active by decision period t. Thus, the state observation of each agent n is given by $o_t^n \in \mathcal{S}^n$, and the system state by $\mathbf{s}_t = (o_t^1 \dots o_t^{N_t}) \in \mathcal{S}$. The joint action of all agents is denoted by $\mathbf{a}_t = (a_t^1 \dots a_t^{N_t}) \in \mathcal{A}$. At decision period t, each agent selects the actions based on a deterministic policy $\pi^n : \mathcal{S}^n \mapsto \mathcal{A}^n$. We let $\pi = (\pi^1 \dots \pi^{N_t})$ define the joint policy. In our particular case, the transition probabilities do not depend on the selected actions, i.e., $\mathcal{P}_t^n(\mathbf{s}_t, \mathbf{a}_t) = \mathcal{P}_t^n(\mathbf{s}_t)$ as we consider a contextual bandit formulation. Thus, the state transition is given by $o_{t+1}^n \sim \mathcal{P}_t^n(\mathbf{s}_t)$, where $\mathcal{P}_t^n \in \mathcal{P} : \mathcal{S} \times \mathcal{S} \mapsto [0, 1]$.

Then, at the end of each decision period t, the agents receive a reward $r_t^n(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{R}_t^n(\mathbf{s}_t, \mathbf{a}_t)$, where $\mathcal{R}_t^n \in \mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. We can then write the performance objective of agent n with policy π^n as an expectation:

$$J^n(\pi^n \mid \pi^{-n}) = \int_{\mathcal{S}} \rho(s)\, r^n(\mathbf{s}, \pi(\mathbf{s}))\, \mathrm{d}\mathbf{s} = \mathbb{E}_{\mathbf{s}\sim\rho}[r^n(\mathbf{s}, \pi(\mathbf{s}))], \tag{4.1}$$

where $\rho(s)$ is the stationary state distribution and π^{-n} is the set of all policies except the one of agent n.

4.4.2. Multi-agent contextual bandit

Let us now particularize the above Markov game formulation for our case. As detailed in Section 4.3, the bit threshold rule operates at each TTI granularity. However, the configuration of this threshold does not need to be done at each TTI. In fact, in the standard O-RAN architecture, the xApps in the Near-RT-RIC make decisions every 100 ms. To be standard compliant, we adopt this time granularity to configure the

bit threshold as a function of traffic load variations. Thus, we make decisions at each decision period t, which comprises a set of TTIs $\mathcal{K}_t$. We associate each BS $b \in \mathcal{B}$ with one learning agent $n \in \mathcal{N}$. Hence, we denote $\mathcal{T}_t^n = \{\mathcal{T}_k^b \mid k \in \mathcal{K}_t\}$ as the set of TBs generated by the BS b corresponding the learning agent n during the decision period t.

The state observation $o_t^n \in \mathcal{S}^n$ for agent n aims to characterize the traffic conditions of the TBs generated by its associated BS. Hence, we define the state $o_t^n = \Phi(\mathcal{T}_t^n, D)$ as the 3-dimensional histogram of the TB's features, i.e., MCS, SNR, and size, where D is the number of bins of this histogram in each dimension. Finally, the action selected by agent n determines the bit threshold of its associated BS, i.e., $a_t^n \in \mathcal{A}^n = [l_{\min}, l_{\max}]$ $\forall n \in \mathcal{N}_t$, where $l_{\min}$ and $l_{\max}$ are the minimum and maximum TB size values.

We now let $E_t(\mathbf{s}_t, \mathbf{a}_t)$ denote the energy consumed by the O-Cloud platform (considering both the CPU resources and the hardware accelerator) during period t.

We also (re-)define the ratio of successfully processed TBs of the O-Cloud as $\zeta_t(\mathbf{s}_t, \mathbf{a}_t)$. Note that both E_t and ζ_t depend on the global state and the joint actions taken, indicating that both the input traffic and the offloading strategy of all the BSs have a joint impact on the performance metrics of the system. Now, using the definition of expectation in equation (4.1), we can define the following constrained decision-making problem:

$$\begin{aligned} \min_{\pi} \quad & \mathbb{E}_{\mathbf{s}\sim\rho}\left[E(\mathbf{s}, \pi(\mathbf{s}))\right] \\ \text{s.t.} \quad & \mathbb{E}_{\mathbf{s}\sim\rho}\left[\zeta(\mathbf{s}, \pi(\mathbf{s}))\right] \geq 1 - \epsilon. \end{aligned} \tag{4.2}$$

We approximate this constrained problem with an unconstrained one by defining the following reward function:

$$r_t(\mathbf{s}_t, \mathbf{a}_t) = -\left(E_t(\mathbf{s}_t, \mathbf{a}_t) + \lambda\left(1 - \zeta_t(\mathbf{s}_t, \mathbf{a}_t)\right)\right), \tag{4.3}$$

where λ is a constant that weights the penalty incurred when failing the constraint and the minus sign converts a cost into a reward to be maximized. Using this reward function in equation (4.1), we define the optimal policy for agent n as

$$\pi_*^n = \underset{\pi^n}{\operatorname{argmin}}\, J^n(\pi^n \mid \pi^{-n}). \tag{4.4}$$

At this point, we would like to remark two practical considerations.

First, the decision periods defined in this section are several orders of magnitude higher than the TTI. Thus, the effect of $\mathbf{a}_{t-1}$ on $r_t(\mathbf{s}_t, \mathbf{a}_t)$ is negligible and therefore each decision period is independent of each other. For that reason, instead of formulating the problem as a multi-agent reinforcement learning (where actions affect future states and rewards), we do it as a multi-agent contextual bandit. This formulation simplifies the problem, allowing us to design a lighter and more effective solution.

Second, the action $a_t^n = \pi^n(o_t^n)$ used by the agent n during the decision period

t is computed based on the distribution of the incoming traffic characterized by the observation o_t^n. Due to the nature of the system, the distribution of the incoming traffic during the decision period t is unknown at the beginning of this decision period. For that reason, we use the observation of the previous decision period $t-1$ to make decisions at t, assuming that traffic distribution changes slower than the time granularity of the decision periods.

4.4.3. Mean field multi-agent solution

A common approach in multi-agent learning is to use the *centralize training with decentralized execution* framework [60, 98] to let actor-critic policy gradient methods include information about other agents. The critic of agent n approximates its reward function and is denoted by $R^n(\mathbf{s}, \mathbf{a} \mid \theta_R^n)$, where θ_R^n are the weights of its function approximator. The loss of the critic is given by:

$$L(\theta_R^n) = \mathbb{E}_{\mathbf{s}_t \sim \rho, a_t \sim \pi', r_t} \left[\left(R^n(\mathbf{s}_t, \mathbf{a}_t \mid \theta_R^n) - r_t(\mathbf{s}_t, \mathbf{a}_t) \right)^2 \right] \tag{4.5}$$

where the expectation considers that the states $\mathbf{s}_t$ follow ρ, the actions are selected with policy π' that can deviate from π (off-policy learning), and the reward samples are noisy.

We denote the actor of agent n as $\pi^n(o^n \mid \theta_\pi^n)$, where θ_π^n are the parameters of the policy. The actors are updated by applying the chain rule to the performance objective defined in equation (4.1) with respect to the actor parameters [99]:

$$\begin{aligned} &\nabla_{\theta_\pi^n} J^n(\pi^n \mid \pi^{-n}) \approx \\ &\mathbb{E}_{\mathbf{s}_t \sim \rho} \left[\nabla_{a^n} R^n(\mathbf{s}_t, \mathbf{a}_t \mid \theta_R^n) \mid_{a^n = \pi^n(o_t^n \mid \theta_\pi^n)} \nabla_{\theta_\pi^n} \pi^n(o_t^n \mid \theta_\pi^n) \right]. \end{aligned} \tag{4.6}$$

Note that the critic needs information about all the agents as $\mathbf{s}_t$ and that $\mathbf{a}_t$ includes the observations and actions of the N_t active agents at decision period t. This limits the scalability of this approach as the critic's complexity increases exponentially with the number of learning agents. Hence, this approach is typically limited to a small number of agents. In our case, we face an additional challenge as the number of active agents changes over time. This limits the applicability of standard function approximators such as feed-forward neural networks that have a fixed input dimensionality.

In order to tackle these challenges, we adopt ideas from Mean Field Theory [61, 100]. To this end, we assume that the interactions within the population of active agents are approximated by the interaction between a single agent and the average effect of the overall population. Hence, the interactions between each agent and a virtual agent approximating the rest of the agents are mutually reinforced. In this way, the optimal policy of an agent is updated based on the behavior of the overall population, while the population of agents is updated based on the individual policies.

Algorithm 1: ECORAN-P

Input: Batch size Z, D^o, D^a
Initialize: Reply buffer $\mathcal{D}^n = \emptyset$, θ_R^n, and $\theta_\pi^n \ \ \forall n \in \mathcal{N}$
for $t = 1 \dots T$ **do**
  Observe the system state $\mathbf{s}_t = (o_t^1 \dots o_t^{N_t})$
  Compute mean field approx. $\bar{\mathbf{o}}_t^{-n} = \Phi(\mathcal{T}_t^{-n}, D^o)$
  Compute the actions $a_t^n = \bar{\pi}^n(o_t^n, \bar{o}_t^{-n} \mid \theta_\pi^n) \ \forall n \in \mathcal{N}_t$
  Use $\mathbf{a}_t$ during t and observe the reward r_t
  Compute mean field approx. $\bar{\mathbf{a}}_t^{-n} = \Phi(\mathbf{a}_t^{-n}, D^a) \ \forall n \in \mathcal{N}_t$
  Store $\langle o_t^n, \bar{\mathbf{o}}_t^{-n}, a_t^n, \bar{\mathbf{a}}_t^{-n}, \mathbf{s}_t, r_t \rangle$ in $\mathcal{D}^n \ \forall n \in \mathcal{N}_t$
  for $n = 1 \dots N_t$ **do**
    Sample Z experiences $\langle o^n, \bar{\mathbf{o}}^{-n}, a^n, \bar{\mathbf{a}}^{-n}, \mathbf{s}_t, r \rangle$ from $\mathcal{D}^n$
    Update θ_R^n by minimizing the critic loss
    $L(\theta_R^n) = \frac{1}{Z} \sum_i \left(\bar{R}^n(o_i^n, a_i^n, \bar{\mathbf{o}}_i^{-n}, \bar{\mathbf{a}}_i^{-n} \mid \theta_R^n) - r_i \right)^2$
    Update θ_π^n by applying the sampled policy gradient
    $\nabla_{\theta_\pi^n} J(\pi^n) \approx \frac{1}{Z} \sum_i \nabla_{\theta_\pi^n} \bar{\pi}^n(o_i^n, \bar{\mathbf{o}}_i^{-n} \mid \theta_\pi^n) \cdot$
    $\nabla_{a^n} \bar{R}^n(o_i^n, a^n, \bar{\mathbf{o}}_i^{-n}, \bar{\mathbf{a}}^{-n} \mid \theta_R^n)$, where
    $a^n = \bar{\pi}^n(o_i^n, \bar{\mathbf{o}}_i^{-n} \mid \theta_\pi^n)$ and
    $\bar{\mathbf{a}}^{-n} = \Phi(\{\bar{\pi}^{n'}(o_i^{n'}, \bar{\mathbf{o}}_i^{-n'} \mid \theta_\pi^{n'}) \mid n' \neq n\}, D^a)$

To this end, we define the mean field critic as $\bar{R}^n(o_t^n, a_t^n, \bar{\mathbf{o}}_t^{-n}, \bar{\mathbf{a}}_t^{-n} \mid \theta_R^n)$, where $\bar{o}_t^{-n}$ is the mean field approximation of the observations of all the agents except n; and, similarly, $\bar{a}_t^{-n}$ is the mean field approximation of the actions of all the agents except n. Specifically, we define $\bar{\mathbf{o}}_t^{-n} = \Phi(\mathcal{T}_t^{-n}, D^o)$, where $\mathcal{T}_t^{-n} = \bigcup_{n' \neq n \in \mathcal{N}_t} \mathcal{T}_t^{n'}$; and $\bar{\mathbf{a}}_t^{-n} = \Phi(\mathbf{a}_t^{-n}, D^a)$, where $\mathbf{a}_t^{-n} = (a_t^1 \dots a_t^{n-1}, a_t^{n+1} \dots a_t^{N_t})$. Moreover, we expand the input of the actor to include the mean field approximation of the observation of the rest of the agents, i.e., $\bar{\pi}^n(o_t^n, \bar{o}_t^{-n} \mid \theta_\pi^n)$, which show better empirical performance. Note that the dimensionality of the input of the mean field critic does not depend on the number of agents, which improves scalability and allows a variable number of learning agents over time. Now, we can redefine equation (4.5) as:

$$L(\theta_R^n) = \mathbb{E}_{\mathbf{s}_t \sim \rho, a_t \sim \pi', r_t} \left[\left(\bar{R}^n(o_t^n, a_t^n, \bar{\mathbf{o}}_t^{-n}, \bar{\mathbf{a}}_t^{-n} \mid \theta_R^n) - r_t(\mathbf{s}_t, \mathbf{a}_t) \right)^2 \right] \tag{4.7}$$

and (4.6) as:

$$\nabla_{\theta_\pi^n} J^n(\pi^n \mid \pi^{-n}) \approx \mathbb{E}_{\mathbf{s}_t \sim \rho} \left[\nabla_{a^n} \bar{R}^n(o_t^n, a_t^n, \bar{\mathbf{o}}_t^{-n}, \bar{\mathbf{a}}_t^{-n} \mid \theta_R^n) \nabla_{\theta_\pi^n} \bar{\pi}^n(o_t^n, \bar{\mathbf{o}}_t^{-n} \mid \theta_\pi^n) \right], \tag{4.8}$$

where

$$a^n = \bar{\pi}^n(o_t^n, \bar{\mathbf{o}}_t^{-n} \mid \theta_\pi^n), \tag{4.9}$$

$$\bar{\mathbf{a}}_t^{-n} = \Phi(\{\bar{\pi}^{n'}(o_t^{n'}, \bar{\mathbf{o}}_t^{-n'} \mid \theta_\pi^{n'}) \mid n' \neq n \in \mathcal{N}_t\}, D^a). \tag{4.10}$$

The pseudo-code of our solution is shown in Algorithm 1.

4.5. Experimental evaluation

Our experimental platform consists of a general-purpose server with an Intel Xeon Gold 6240R CPU with 16 cores dedicated to signal processing tasks and an NVIDIA GPU V100. The O-Cloud's Acceleration Abstraction Layer (AAL) [44] is implemented using Intel Data Plane Development Kit (DPDK) Baseband Device Library (BBdev). To process 5G signals, we use a publicly available software library from Intel (Intel FlexRAN [87]) and a proprietary driver for the GPU. The Near-RT-RIC comprises an Intel i7-8750H CPU @ 2.20GHz and 8 Gb of Random Access Memory (RAM).

We have integrated ECORAN in a real O-RAN platform. In particular, ECORAN-P is hosted in the Near-RT-RIC as a xApp and computes the bit threshold for each BS. On the other hand, ECORAN-P is implemented into each BS and enforces this bit threshold offloading policy at the TB level. Figure 4.4 shows a scheme of our experimental platform.

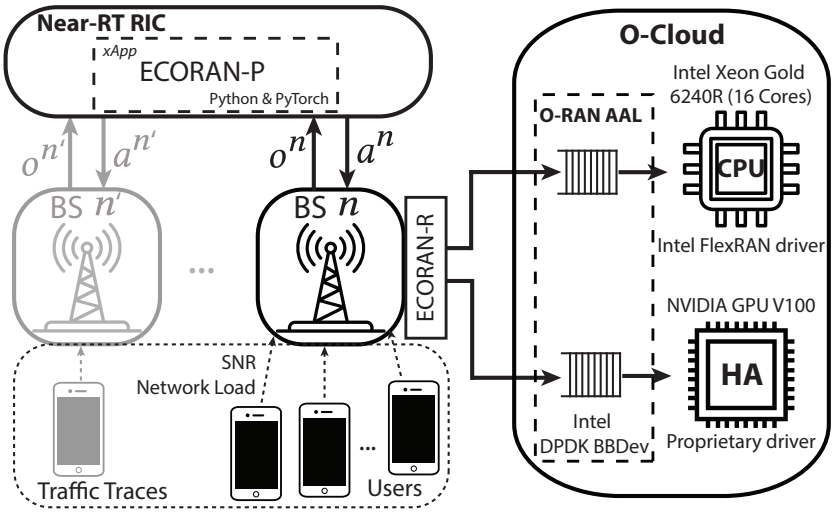


Figure 4.4: Scheme of our experimental platform. Each BS has connected a set of mobile users that generate traffic load. All the BS exchange information with the near-RT RIC. Every t, the BSs send their observations o_t^n and receive their respective action a_t^n. For each TB, the BSs decided whether to use software processing or the HA.

To emulate real 5G processing workloads, we collected traces from real base stations in Madrid, Spain, during April 2022 using an open-source tool called Falcon [81]. Based

on these traces, we emulate the workload generated by two types of BS, as shown in Figure 4.5. The left plot depicts the empirical distribution of the size of each TB generated; the center plot illustrates the dynamics of the rate of TBs; and the right plot depicts the SNR distribution of both BSs. In this way, we emulate two distinct BSs, a BS with a single-yet-demanding user (Type 1) and a BS with 10 lightly-loaded users (Type 2), with realistic network demand and mobility patterns.

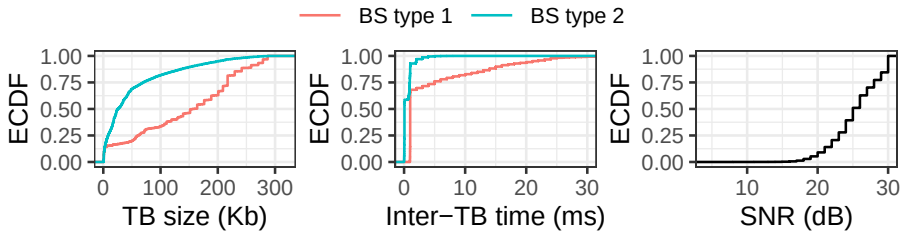


Figure 4.5: Characterization of the incoming traffic to each of the BS types considered.

In the rest of this section, we combine these two types of BSs to generate heterogeneous traffic loads for our experiments. In particular, in the settings with 6, 12, 36, 48, 60 active BS, there are 5, 10, 30, 40, 50 BSs of Type 1, respectively, and the rest of Type 2. Although we assess two types of BS, the instantaneous load of individual BSs is very bursty due to the trace-based mobility patterns and demands we emulate. Consequently, each BS requires individual offloading strategies.

For the implementation of ECORAN, both actor and critic have a similar neural network architecture with two 3D-convolutional layers with 8 and 1 cells, respectively, and a kernel of size 3. These convolutional layers are connected to 2 fully-connected layers of 256 units per layer. The output of the actor and critic are activated with sigmoid and linear functions, respectively. Figure 4.6 shows the architecture of one learning agent.

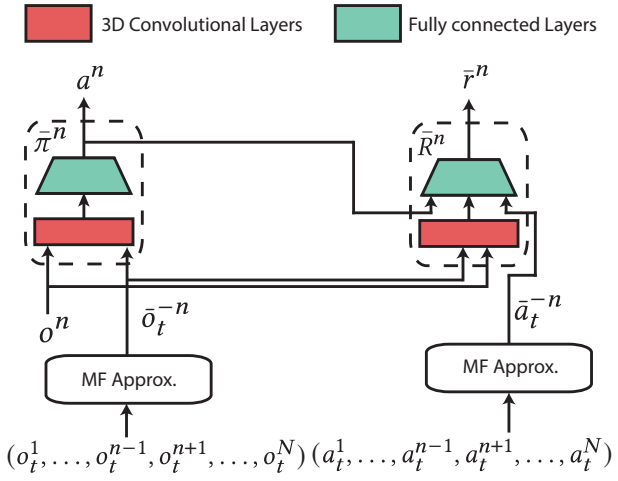


Figure 4.6: Scheme of the architecture of learning agent n.

Note that only the data represented with 3-dimensional matrices is fed into the 3D convolutional layers, while the rest of the data (vector representation) is fed into the critic's fully connected layers. The noise used for exploration follows an Ornstein-Ulhenbeck process with parameters $\theta_{\text{noise}} = 0.15$ and $\sigma_{\text{noise}} = 0.15$, generating temporally correlated values around 0 [101]. Since all the learning agents share the same goal, we consider the particular case in which the reward signal is common and the weights of the neural networks are shared across all the agents, which is a common practice in similar settings [61, 102–104]. We configure the algorithm with a batch size of 64 samples and $D = D^a = D^o = 5$. We empirically found that $\lambda = 2$ is large enough to avoid solutions that violate the reliability constraint. The minimum and maximum values of the TB size ($l_{\text{min}} = 32$ bits, $l_{\text{max}} = 286976$ bits) are given by the 3GPP standard organization [92].

4.5.1. Convergence evaluation

To evaluate the convergence of ECORAN, we consider a scenario with 36, 48, and 60 BSs (i.e., learning agents). This is in line with current large-scale deployments implemented by mobile operators [33]. Figure 4.7 shows the evolution of the reward during the training phase. While the complexity of typical multi-agent methods (e.g., [60, 105]) grows exponentially with the number of agents, we show in Figure 4.7 that our solution does not show such a dependency. The reason is twofold. First, we share neural network weights across agents, and therefore the weight updates of each agent benefit all others. Second, for every t, we generate as many experience tuples (see line 11 of Algorithm 1) as the number of learning agents. A larger amount of data benefits the learning rate (off-policy learning). These two aspects compensate for the increase in complexity to coordinate a larger number of agents, that is in line with the cost-effective design goal of centralizing multiple BS (agents) in the same shared computing platform.

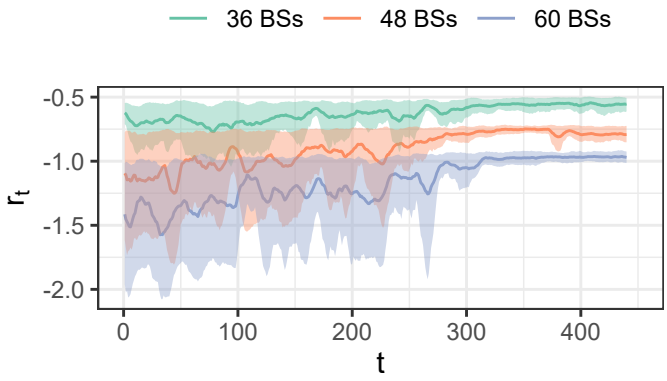


Figure 4.7: Reward of the system during training for 36, 48, and 60 learning agents. The mean and 10^{th}, and 90^{th} percentiles (colored shade) of 10 independent runs are shown.

4.5.2. Comparison with other methods

We now compare ECORAN with a number of benchmarks:

- **Dedicated HAs.** This is the industry standard approach: each BS offloads every TB to a dedicated HA to guarantee reliability. We emulate as many HAs as BSs.
- **Always Offload.** BSs offload all their TBs to a shared HA.

All the remaining benchmarks rely on an O-Cloud with two shared processing resources: a HA and a CPU pool.

- **Minimum Waiting Time (MWT).** For each TB, the BS selects the computing resource that minimizes the total waiting time, i.e., $a_i = \text{argmin}_a \ \gamma(a, d_i) + q(a, d_i)$. This is a common approach in queuing theory aimed at minimizing latency. Note that, to evaluate this strategy, the processing times given by $\gamma(\cdot)$ should be known in advance, which is not possible in a real system. Hence, we use a dataset with processing times for all possible cases and then we simulate the system based on these pre-known times.
- **ECORAN$_{\text{-MF}}$.** This approach uses ECORAN *without* the mean field approximation, i.e., $\bar{\mathbf{o}}_t^{-n}$ and $\bar{\mathbf{a}}_t^{-n}$ are not available. Thus, each agent becomes an independent learner that only uses its own observations and actions for learning.
- **ECORAN$_{\text{CLDE}}$.** This is ECORAN with a *Centralized Learning Decentralized Execution (CLDE)* approach [60, 105] instead of the mean field approximation. This approach does not support a changing number of agents during learning or execution. We circumvent this challenge heuristically by overdimensioning the algorithm to the maximum number of agents in our training and evaluation, and filling with zeros the observations and actions of the agents that are not active.

We train ECORAN (and its variants) for a general scenario with a changing number of BSs (up to 60). Note that due to the integration of the mean field approximation in our solution, the number of learning agents can vary during both training and execution. After training, we evaluate the same instance of our algorithm in several scenarios with a different number of active BSs. In this way, we also assess the ability of our approach to *adapt* to any trivial deployment. Figure 4.8 shows the energy cost (left), the percentage of bits that are timely decoded (center), and the share of TBs that are offloaded to a HA (right). Moreover, Table 4.1 details the energy savings of each approach compared to using dedicated HAs (the industry standard), and the target reliability gap.

"Always Offload" is one of the strategies with the highest energy consumption. However, because the HA is shared across all the BSs and the CPU resources are not exploited, this strategy is not able to meet the reliability target during high-load scenarios, which is a hard requirement in our system. In the case of "MWT", the computing resource with the lowest total waiting time is selected. During low-load scenarios (i.e., 6 and 12 BSs) this strategy frequently selects the HA (86% and 90%, respectively, on average), which consumes substantial energy. Moreover, such a strategy leads to saturating the HA

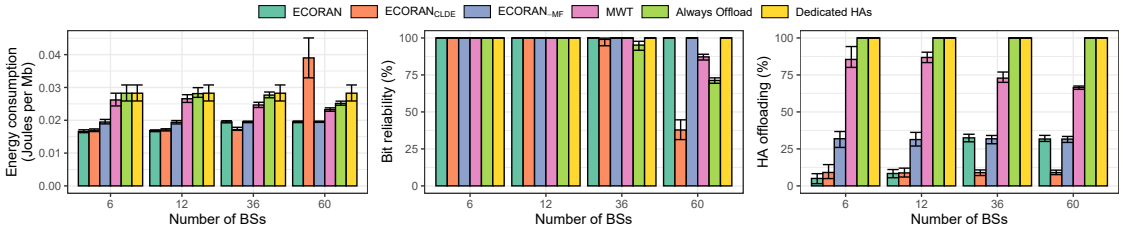


Figure 4.8: Energy consumption in Joules per Mb (left), bit reliability or percentage of correctly decoded bits (center), and HA offloading or percentage of TBs processed into the HA (right). Each bar shows the median, 10^{th}, and 90^{th} percentiles.

	Energy Savings (%)				**Reliability gap (%)**			
# BSs / **Approach**	6	12	36	60	6	12	36	60
ECORAN	**41.23**	**40.51**	30.96	**31.00**	**0**	**0**	**0**	**0**
ECORAN$_{\text{CLDE}}$	40.03	39.69	**38.75**	-38.94	**0**	**0**	2.01	62.02
ECORAN$_{\text{-MF}}$	31.03	31.48	31.02	30.92	**0**	**0**	**0**	**0**
MWT	6.97	5.91	12.75	17.74	**0**	**0**	0.08	12.96
Always Offload	0	0	1.95	10.99	**0**	**0**	5.23	28.72

Table 4.1: Performance comparison. Energy savings per bit computed w.r.t dedicated HA (industry standard). The reliability gap shows the percentage of extra reliability needed to meet industry-grade reliability, i.e., 99.999%.

resource, which is fatal for high-load scenarios as the CPU is often forced to process large TBs, which renders poor reliability, up to 12% below the target value. Conversely, we can observe that, like when using dedicated HAs, only ECORAN and ECORAN$_{\text{-MF}}$ meet the target reliability in all of the scenarios. However, compared to using dedicated HAs, ECORAN achieves so with up to 41% energy savings.

Let us discuss why ECORAN$_{\text{CLDE}}$ and ECORAN$_{\text{-MF}}$ obtain suboptimal solutions. The former minimizes energy consumption but violates the reliability target in high-load scenarios, and the latter meets the constraint but consumes more energy with lower traffic loads. In ECORAN$_{\text{-MF}}$, both actor and critic ignore the information about other agents because the mean field approximations are not available. Thus, the agents cannot distinguish whether the number of active BSs is low or high (small or large workloads). Thus, they are unable to learn the relationship between the reward signal (common to all agents) and the actual system state. Conversely, the critic in ECORAN$_{\text{CLDE}}$ has a global view of the system at the cost of extra complexity. Although it can potentially learn the relationship between the global state and the reward during centralized learning, the actor only sees local observations, limiting adaptability during decentralized execution. In contrast, the mean field strategy adopted in ECORAN allows the algorithm to have a global view of the system without compromising complexity and scalability as both the global state and the global action are characterized by vectors of fixed dimensionality independent of the number of active agents. With 60 agents (BS) being the maximum

tested in our evaluation, ECORAN achieves up to 60x cost gains compared to the industry standard approach, which deploys as many HAs as BSs. Notably, ECORAN operates with only one HA assisting all the BSs, which further highlights its cost-effectiveness.

4.5.3. System validation

We now evaluate the practicality of our solution. We have two requirements for the xApp hosting ECORAN-P in the Near-RT-RIC: (*i*) the execution time needs to be faster than the time granularity of 100 ms of the system update; and (*ii*) produce a negligible energy footprint. Note that a heavy model can be either too slow to operate in real-time or can offset the energy savings if deployed in energy-hungry hardware.

Table 4.2 (left) shows the time and energy burden of ECORAN-P executed in the Near-RT-RIC. We consider the worst-case scenario with 60 agents for execution and a model update (actor and critic) for training. We observe an execution time shorter than 1 ms with a negligible energy burden. Conversely, although the training task is more demanding in terms of time and energy, it is only performed temporarily and can even be executed in the background on another machine if needed (off-policy training).

Table 4.2 (right) shows the percentage of energy consumed by ECORAN-P over the total energy consumed by the Near-RT-RIC and the O-Cloud together. For that, we consider a worst-case scenario in which, apart from the action computation, ECORAN-P is trained during 30 s (convergence time in Figure 4.7) every hour. Moreover, we consider low (6 BSs) and high (60 BSs) traffic loads, as the energy consumption of the O-Cloud is highly dependent on the load. We measure an energy footprint of 3.2% in the worst case and below 1% when the O-Cloud has a medium or high load. This validates the practicality of our solution, which operates in a real O-RAN system meeting its time constraints and providing real energy and cost savings compared to the industry standard.

	Time (ms)	**Energy (J)**	**Traffic Load**	**% of Energy**
Execution	0.572	0.008	**Low**	3.203 %
Training	46.267	2.312	**High**	0.390 %

Table 4.2: Average time and energy burden of ECORAN-P executed in the near-RT RIC with an Intel i7-8750H CPU. The percentage of energy consumed by ECORAN-P over the total energy (O-Cloud plus near-RT RIC) for different traffic settings in the O-Cloud.

5 Multiplexing GPU resources in GPU-accelerated vRANs

In line with the analysis carried out in Section 3.4, reusing spare resources of under-utilized Hardware Accelerators (HA) to handle workloads that are unrelated to the Distributed Unit (DU) processing is a promising direction for building more cost-effective virtualized Radio Access Network (vRAN) systems compared to solutions that dedicate individual HAs to individual HA-based tasks. As already discussed in Section 1.1, Graphics Processing Units (GPU) have been proposed as a realistic candidate for HA thanks to their unique capability of handling GPU-based Machine Learning (ML) models that can be used to improve and automate DU operations.

However, this chapter shows that multiplexing the available resources of under-utilized GPUs by blindly co-locating DU workloads and ML services can compromise the processing reliability of the former and the throughput performance of the latter. More in general, GPU multiplexing comes with many challenges that need to be properly considered at the time of designing effective solutions. Under this perspective, this chapter presents YinYangRAN, an innovative system operating in the Non-Real-Time (Non-RT) RAN Intelligent Controller (RIC) that supervises the multiplexing of the computing resources of a single GPU-based HA as to ensure reliability in processing DU tasks while maximizing the throughput of a concurrent ML service running in the same GPU.

5.1. Typical challenges in GPU multiplexing

Online multiplexing of spare resources in GPU-based HAs between virtualized DUs and third-party applications running ML models can enhance the cost-efficiency of vRANs. This strategy effectively amortizes the high costs of these expensive processors and eliminates the need to dedicate individual HAs to individual HA-based tasks. While this approach has the potential to yield significant cost savings, GPU multiplexing between DU processing and ML also comes with its own set of challenges, which will be discussed in the next subsections.

5.1.1. Reliability in traditional GPU multiplexing schemes

State-of-the-art multiplexing mechanisms do not provide viable brokering solutions to guarantee industrial target reliability values in the vRAN. In fact, Figure 5.1 (dashed red curve) shows that co-locating a realistic ML inference task with the DU workloads in an A100 GPU using the conventional software concurrency scheme presented in Section 2.3 results in a substantial disruption of the 5G pipeline. Indeed, the DU processing latency becomes highly erratic, with a low 50% probability of meeting latency targets, that is an unacceptable industry-grade performance.

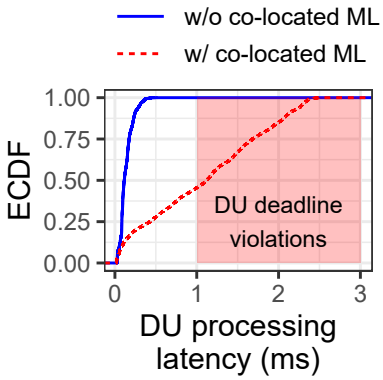


Figure 5.1: Empirical CDF of 5G DU processing latency.

In light of these results, it does become evident that GPU-based vRAN acceleration introduces new resource control challenges that are alien to traditional radio systems, or to conventional vRANs based on Application-Specific Integrated Circuits (ASIC) and Field Programmable Gate Arrays (FPGA) HAs. Specifically, to maximize cost gains, it is crucial to devise robust mechanisms that aptly allocate GPU resources between *inelastic* Physical layer (PHY)-layer processing at DUs and *elastic* ML workloads.

5.1.2. Dependency of DU processing with GPU resources

Considering the case of GPU-based HA, the amount of computing resources (i.e. Streaming Multiprocessors (SM)) assigned to the processor remarkably impact DU processing latency performance. To shed light on the GPU resource requirements by focusing on 5G Forward Error Correction (FEC) decoding, Figure 5.2 presents latency results for three different GPU configurations where 15%, 50% and 100% of the available SMs are reserved to the decoding function. As already observed in previous figures of Subsection 3.3.1, a higher Modulation and Coding Scheme (MCS) typically results in increased latency because it encodes more bits into the Transport Block (TB), thereby requiring a longer decoding time. For a given MCS, a lower Signal-to-Noise Ratio (SNR) can further inflate latency, as the decoder may necessitate more iterations over the data

to decode it. Moreover, reducing the allocation of SM computational resources naturally increases the processing latency too. These results show that it is not trivial to assign computing resources (SMs) to the DU to guarantee with certain probability that its processing latency stays below typical targets imposed by industry.

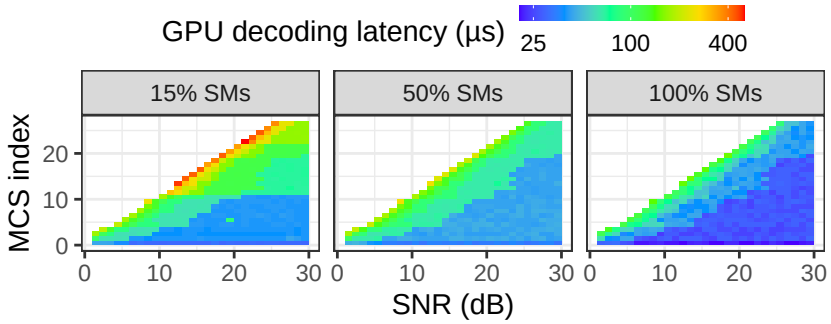


Figure 5.2: Latency performance to process a single 100-MHz TB on an NVIDIA A100 GPU with different combinations of MCS, SNR, and allocation of GPU SM resources.

5.1.3. Overheads in GPU resource re-configuration

Adapting dynamically to real-world DU workloads requires fast enough GPU resource re-configurations. For this reason, it is important to characterize the overheads incurred by state-of-the art GPU resource re-configuration mechanisms, as they are an important factor when taking decisions on updating the share of computing resources of the HA allocated to the DU and to other co-located workloads.

In Section 2.3, we introduced two methods for sharing GPU resources: Multi-Instance GPU (MIG) and Multi-Process Sharing (MPS). As mentioned therein, MIG provides full resource isolation among tasks, while MPS only partitions SMs by keeping memory and caches as shared resources. The two approaches induce very different latency overheads when re-configuring a GPU slice, as reported in Figure 5.3. The plot summarizes 200 random re-configurations on an NVIDIA A100 GPU with both MIG (red) and MPS (blue): each re-configuration changing the partitioning of SMs across tasks, which implies tearing down the existing SM configuration and creating a new one. It is worth noting that during the re-configuration process the GPU cannot run any task.

As shown in the figure, the median latency is fairly constant for each strategy. More precisely, re-configuring a MIG partition in the target GPU takes 6.9 seconds, against the 0.266 seconds of re-allocating SM resources using MPS. During such re-configuration intervals, we let the higher-priority DU fall back to software processing, e.g., using available CPU cores in the cloud, which can process reliably the workload in very frequent scenarios (see Subsection 3.3.1). As continued and substantial performance drops during several seconds are hardly acceptable in production-grade systems, MIG does not seem a

viable option for GPU-based HA multiplexing. On the other hand, the delays introduced by MPS appear bearable for Non-Real Time O-RAN operations that occur at the order of seconds and for a faster adaptation to dynamic DU workloads.

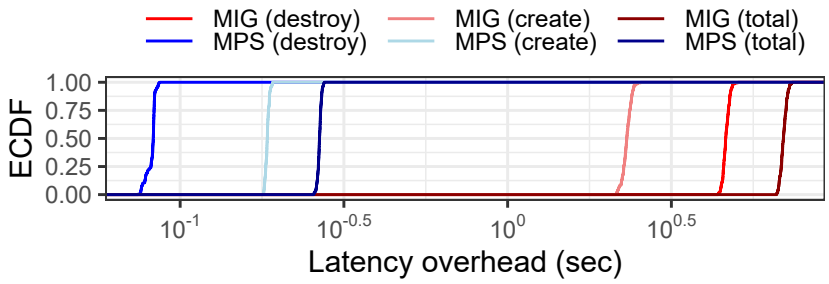


Figure 5.3: Latency overhead when creating, destroying, and re-configuring (creating+destroying) a GPU slice using MIG *vs* MPS technology. Logarithmic x-axis.

We recall that MPS does not provide memory or cache isolation, which could potentially cause resource contention and lower raw performance than MIG. Yet, unlike MIG that is constrained to a small set of possible partition configurations, MPS provides full flexibility when allocating SM resources, which can compensate for the shared memory with a better adaptation to DU workload dynamics.

In conclusion, based on this experimental analysis, we advocate for a dynamic MPS-based GPU resource allocation method, and adopt this strategy in the rest of the chapter.

5.1.4. Trade-offs in DU/ML GPU multiplexing

The industry imposes PHY processing latency targets that range from 0.5 to 3 ms, which the DU must satisfy with a certain probability to attain a certain reliability goal. In order to optimize vRAN cost-efficiency, it is critical to allocate to DUs just the right amount of GPU resources needed to meet their target, so as to free as much computing capacity as possible to maximize ML performance and GPU utilization. The conflicting goals create a trade-off of fulfilling 5G processing latency targets and maximizing the ML throughput. We experimentally characterize this trade-off by analyzing a commercial solution for GPU-accelerated 5G processing on a NVIDIA A100 GPU with 40 GB of Random Access Memory (RAM). Motivated by real-world RAN workloads (see Section 3.1), we consider a wide range of workload contexts, i.e., combinations of MCS, Resource Block (RB), SNR, and concurrent users in a 5G 100-MHz DU. We also run ML tasks on the same GPU and measure the related throughput as the number of inferences per second. More specifically, we deploy the TES-RNN model [5], a state-of-the-art predictor in mobile traffic forecasting that blends statistical modeling and deep neural networks. All this is done with the control of the GPU computing resources (SMs) via MPS by assigning a given amount of resources to the DU and its complementary value

to the ML workload. The result of these experiments is illustrated in Figure 5.4. The plot depicts in blue the FEC processing latency of the DU under diverse GPU resource allocations, along the abscissa. Specifically, the dark blue lines show the highest and lowest latency performance recorded across all possible contexts; the light blue area in between illustrates the possible operating region of the DU for any intermediate context, with thin lines therein showing the latency profiles of some sample contexts. For illustration purposes, the dashed horizontal line highlights one of the most conventional PHY-layer processing latency targets, i.e., 1 ms, which the DU must meet.

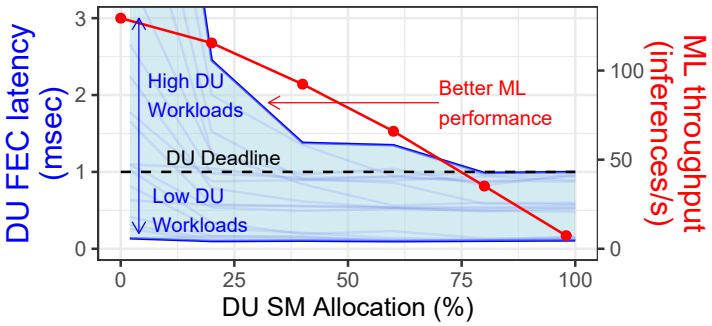


Figure 5.4: DU performance (in blue) *vs* ML performance (in red) for different SM allocations and DU contexts. The x-axis indicates the allocation for DU workloads. The allocation for ML workload is the complementary of the x-axis ((100-x)%). The light blue area depicts the operating region for DU FEC workloads across a wide range of contexts.

We observe that the latency curves span very diverse values, both above and below the 1-ms target. Most relevantly, the minimum fraction of GPU SMs required to fulfil the target ranges all the way from 1% to 80% depending on the DU context. In other words, our experiments demonstrate for the very first time that sub-6GHz 5G PHY-layer processing only needs a (potentially very small) portion of the full capacity of a modern GPU, hence there exists a substantial space for multiplexing DU requests with other ML workloads. To prove this point, as explained above, we also run a trained ML model on the SMs of the same GPU not allocated to the DU processing. The red curve in Figure 5.4 shows that the throughput of the ML model, in terms of inferences per second, is strongly dependent on the amount of spare SMs it is allowed to use. Thus, operating DUs only with the limited GPU resources they really necessitate opens the way to significant gains in terms of ML performance for co-located edge and O-RAN services.

5.2. GPU-accelerated vRAN scenario

We consider an O-Cloud platform comprising one GPU that runs two services at the same time: 5G PHY operations offloaded from a virtualized DU, and a Deep Learning (DL) model offloaded from a ML service, as shown in Figure 5.5.

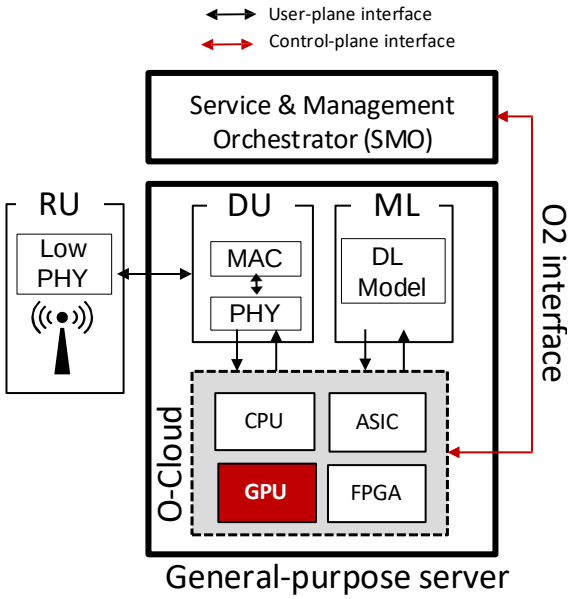


Figure 5.5: GPU-accelerated vRAN scenario.

We use MPS to allocate GPU resources (SM) to each of the services. At each Transmission Time Interval (TTI), a set of TB arrives at the DU to be decoded. Each TB is characterized by its SNR, MCS, and TB size (number of bits). Note that the computational capacity needed by the DU to decode these TBs depends on the number of received TBs that changes at every TTI and the SNR, MCS, and TB size of each TB (see Figure 5.2). Smaller SM allocations for the DU may imply that some TBs are not decoded in time and therefore are discarded. However, once the DU has enough SM resources to decode all the TBs in time, there is no further benefit in assigning more SM resources to this end, which describes an *inelastic* service. Conversely, co-located ML services, which are competing for the same GPU resources, can improve throughput performance if more SM resources were allocated, which describes an *elastic* service.

5.3. Problem formulation

Following the standard O-RAN architecture (see Figure 5.5), we operate in the Service & Management Orchestrator (SMO) with a time granularity $\delta = 1$ second or higher, where $t = 0, 1, 2 \ldots$ denotes the decision periods. The normalized SM allocation for the DU at t is denoted by $a_t \in \mathcal{A} \subseteq [0, 1]$, where $\mathcal{A}$ is a discrete set with all possible allocations; and the set of TBs received by the DU during period t is denoted by $\mathcal{T}_t$.

Now, we let $\phi_t = \Phi(\mathcal{T}_{t-1}, D)$ be a 2-dimensional histogram of the DU contexts (TBs per TTI: SNR and size) that characterize the observed traffic, where D is the number of histogram bins in each dimension. Consequently, we define the system state as $s_t = (\phi_t, a_{t-1}) \in \mathcal{S}$, where $\mathcal{S}$ is the state space.

At each decision period t, an SM allocation a_t is selected. Our goal is to minimize the allocation of SM resources to DU workloads to maximize the performance of the co-located ML service. If configuration a_t differs from the one selected in the previous decision period, i.e., if $a_t \neq a_{t-1}$, the GPU needs to be re-configured, which, as mentioned above, causes ML service disruption during a re-configuration period of duration r.

We model this re-configuration cost as:

$$\Delta(s_t, a_t) = \begin{cases} \frac{r}{\delta} \cdot (1 - a_t) & \text{if } a_t \neq a_{t-1} \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

where δ corresponds to the duration of a decision period t.

Note that the cost acts as a proxy for the throughput of the ML application, following an inverse correlation with it. Therefore, we model the overall system cost at period t as

$$C(s_t, a_t) = a_t + \Delta(s_t, a_t), \tag{5.2}$$

We now denote the ratio of TBs timely processed before their latency target at t by $\zeta(s_t, a_t)$, also referred to as *reliability*. Then, we formulate our problem as a finite horizon Markov Decision Process (MDP) with constraints.

$$\begin{aligned} &\min_{a_0,\ldots,a_{T-1}} J(s_0; a_0, \ldots, a_{T-1}) \\ &\text{s.t.} \quad \gamma^{\tau}\left[\zeta_t(s_t, a_t)\right] > 1 - \epsilon, \qquad \text{for } t = 1, \ldots T \end{aligned} \tag{5.3}$$

where

$$J(s_0; a_0, \ldots, a_{T-1}) := \mathbb{E}\left\{ C_T(s_T, a_T) + \sum_{t=0}^{T-1} C_t(s_t, a_t) \right\} \tag{5.4}$$

is the cost of a sequence of actions $\{a_0, \ldots, a_{T-1}\}$ and an initial state s_0. In addition, $C_T(s_T, a_T)$ is the termination cost, a_T is the termination action, $\gamma^{\tau}[Z]$ provides the τ-quantile of a distribution Z, and ϵ sets the reliability target. Note that the reliability of the system denoted by $\zeta(\cdot)$ is random due to the intrinsic stochasticity of the radio access network. By adjusting the value of τ, we can balance the trade-off between constraint satisfaction probability and cost. In this way, with this formulation, we capture all the important aspects of our problem: (*i*) the cost of the different SM allocations for the ML application; (*ii*) the critical reliability constraint of the DU; and (*iii*) the impact of the re-configurations on the long-term performance of the system.

To obtain the minimum cost and consequently the optimal sequence of actions, we rely on the principle of optimality [106], which states that the optimal sequence of actions for a truncated tail sub-problem (e.g., from t' to T) is also optimal for the full problem.

Therefore, the optimal cost $J^*(s_t)$ for a given state s_t can be obtained recursively as follows:

$$J_t^*(s_t) := \min_{a \in \mathcal{A}} \ \mathbb{E}\left\{C_t(s_t, a) + J_{t+1}^*(s_{t+1})\right\} \qquad (5.5)$$
$$\text{s.t.} \qquad \gamma^\tau \left[\zeta_t(s_t, a_t)\right] > 1 - \epsilon \qquad \text{for } t = 1, \dots T.$$

where $J_T^*(s_T) := C_T(s_T, a_T)$.

Note that the optimal cost cannot be computed in practice as defined above because future traffic conditions $\phi_{t'}$ for $t' > t$ are unknown in advance. In the next section, we present a control strategy to overcome this limitation.

5.4. Algorithm design

We present two algorithms in this section. YinYangRAN-Full, an approximate solution to (5.3) rooted in certainty equivalent control principle (Subsection 5.4.1); and YinYangRAN-Lite, a simplified version of YinYangRAN-Full suitable when $r/\delta \to 0$ (Subsection 5.4.2). Finally, in Subsection 5.4.3, we present a solution to ensure that the lowest quantiles of the DU reliability performance are above the required target.

5.4.1. YinYangRAN-Full (YYR-Full)

To solve the problem defined in equation (5.3), we propose an approximate control strategy, named YinYangRAN-Full (YYR-Full), that relies on Certainty Equivalent Control (CEC) [107]. Our solution comprises two phases. In an offline phase, we replace the uncertain metrics (in our case, future traffic demands) with their estimated values. Thus, based on the CEC principle, the problem in equation (5.3) becomes deterministic and we can compute an estimation of the optimal cost. Second, during online operation, YinYangRAN-Full sequentially selects an action based on system observations m and the estimation of the optimal cost computed in the offline phase.

5.4.1.1. Offline phase

We define $\tilde{\phi} = \{\tilde{\phi}_0 \dots \tilde{\phi}_{T-1}\}$ as the sequence of expected traffic demands. Using $\tilde{\phi}$ in equation (5.5), we compute an estimation of the optimal cost as follows:

$$\tilde{J}_t(\tilde{s}_t) := \min_{a_t \in \mathcal{A}} \ \mathbb{E}\left\{C(\tilde{s}_t, a_t) + \tilde{J}_{t+1}(\tilde{s}_{t+1})\right\} \qquad (5.6)$$
$$\text{s.t.} \qquad \tilde{\zeta}_t(\tilde{s}_t, a_t, \tau) > 1 - \epsilon \qquad \text{for } t = 1, \dots T$$

where $\tilde{J}_T(\tilde{s}_T) := \Delta(\tilde{s}_T, a_T)$, $\tilde{s}_t := (\tilde{\phi}_t, a_{t-1})$, and $\tilde{\zeta}_t(\tilde{s}_t, a_t, \tau)$ is an approximation of the τ quantile of the reliability for the pair $(\tilde{s}_t, a_t)$. The latter can be obtained, for instance,

by training a neural network with a quantile loss as we detail in Subsection 5.4.3. The termination cost is defined as the re-configuration cost of a predefined termination action a_T. To accomplish this phase, we use equation (5.6) to compute the estimation of the optimal cost for all possible values of $\tilde{s}$, i.e., $(T+1) \times |\mathcal{A}|$ values in total, which are subsequently stored in memory. More formally, the approximation of the optimal cost detailed in equation (5.6) has a complexity of $O(T \times |\mathcal{A}|^2)$. Note that, even when we consider a large T, this phase is computed offline and hence its execution time is not a limitation. Nevertheless, if the offline execution time is a limitation for large T values, we propose in Subsection 5.4.2 a lighter version, which ignores the re-configuration costs and is suitable when the re-configuration costs are negligible, i.e., when $r/\delta \to 0$.

5.4.1.2. Online phase

During online operation, YYR-Full selects each a_t solving a one-step look-ahead problem:

$$\begin{aligned} \min_{a_t \in \mathcal{A}} \quad & C(s_t, a_t) + \tilde{J}_{t+1}(\tilde{\phi}_{t+1}, a_t) \\ \text{s.t.} \quad & \hat{\zeta}(s_t, a_t, \tau) \geq 1 - \epsilon. \end{aligned} \tag{5.7}$$

Note that the online phase of YYR-Full uses s_t, which includes actual DU traffic load observations ϕ_t. Thus, the current cost is computed accurately according to system observations, and using $\tilde{J}_{t+1}(\cdot)$ we estimate the impact of current actions in the future cost. These two terms provide a computationally efficient farsighted decision-making mechanism. We make two remarks. First, the CEC strategy obtains optimal results when the future states match their estimations. However, when the observed traffic conditions differ from their expectation, this strategy still provides good empirical results as we show later in this thesis. Second, equation (5.7) can be solved with low complexity as it only requires searching over the action space and therefore its complexity is $O(|\mathcal{A}|)$. This makes YYR-Full suitable for online operation in real systems.

5.4.2. YinYangRAN-Lite (YYR-Lite)

Due to the re-configuration cost of our problem, it is very important for our control strategy to be farsighted. An over-adaption to the changes in the traffic load can lead to a large number of re-configurations, whose cost should be taken into account in the design of an efficient control scheme. However, in some cases, the re-configuration cost can be considered negligible. For example, if the duration of each decision period δ is in the order of minutes; or when the re-configuration time of the GPU r is very small. In these cases, the impact of the future cost on the decision-making tends to zero and we can rely on a simpler version of YYR-Full.

In such a case, at every decision period t, the objective is to select greedily the minimum SM allocation for the DU a_t that satisfies the reliability constraint, that is,

$$\begin{aligned} &\min_{a_t \in \mathcal{A}} && a_t \\ &\text{s.t.} && \hat{\zeta}(s_t, a_t, \tau) \geq 1 - \epsilon. \end{aligned} \tag{5.8}$$

Note that with this approach, which we name YinYangRAN-Lite (YYR-Lite), the offline phase used by YYR-Full can be omitted as we rely on a greedy version of the online phase.

5.4.3. Reliability satisfaction scheme

A critical aspect of our problem is reliability satisfaction because the computing workloads of the DU are inelastic. The challenges of reliability satisfaction are deeply analyzed in Section 5.1.4. In addition, as we operate in time scales of one second or longer, the observed values of reliability for a given traffic load and MPS configuration are stochastic, due to the intrinsic randomness of the Radio Access Network (RAN) (due to, e.g., mobility of the users, app-dependent traffic generation, and so on). Taking this into consideration, the goal is to minimize the probability of missing DU latency targets by learning the quantiles of the reliability function. Thus, we can formulate an SM allocation problem that ensures that the lowest quantiles of the distribution are above the required reliability target.

To this end, we define $F_Z(z)$ as the Cumulative Distribution Function (CDF) of Z. For a given quantile $\tau \in [0, 1]$, the value of the quantile function is defined as $q_\tau = F_Z^{-1}(\tau)$. The quantile regression loss is an asymmetric convex function that penalizes overestimation error with weight τ and underestimation error with weight $1 - \tau$:

$$\mathcal{L}^\tau(\hat{q}_\tau) := \mathbb{E}_{z \sim Z}\left[\rho_\tau(z - \hat{q}_\tau)\right], \text{ where} \tag{5.9}$$

$$\rho_\tau(u) := u \cdot (\tau - \Gamma_{\{u<0\}}) \quad \forall u \in \mathbb{R} \tag{5.10}$$

where $\hat{q}_\tau$ is the quantile function estimation, and $\Gamma_{\{x\}}$ takes value 1 when the condition x is met and 0 otherwise. We let the reliability estimator $\hat{\zeta}(\cdot)$ have N outputs, which approximates the set $\{q_{\tau_1}, \ldots, q_{\tau_N}\}$. Thus, we train $\hat{\zeta}(\cdot)$ using stochastic gradient descent to minimize the following joint objective:

$$\sum_{i=1}^{N} \mathcal{L}^{\tau_i}(\hat{q}_{\tau_i}). \tag{5.11}$$

It is worth mentioning that the quantile regression loss presents a discontinuity at zero,

which limits its practical performance when using function approximators such as Neural Networks (NN). To overcome this limitation, we consider the quantile Huber loss [108]. This loss function has an asymmetric squared shape in an interval $[-\kappa, \kappa]$, and reverts to the standard quantile loss outside of this interval:

$$L_\kappa(u) := \begin{cases} \frac{1}{2}u^2 & \text{if } |u| \leq \kappa \\ \kappa(|u| - \frac{1}{2}\kappa) & \text{otherwise.} \end{cases} \tag{5.12}$$

Thus, the asymmetric variation of the Huber loss is

$$\rho_\tau^\kappa(u) := |\tau - \delta_{\{u<0\}}| \frac{L_\kappa(u)}{\kappa}. \tag{5.13}$$

Finally, the quantile Huber loss can be derived by introducing $\rho_\tau^\kappa(u)$ in eq (5.9). Note that when κ tends to zero the quantile Huber loss reverts to the quantile regression loss.

5.5. System implementation

We implemented an O-RAN system as depicted in Figure 5.6. The O-Cloud server is an High Performance (HP) server with an Intel Xeon Gold 6240R Central Processing Units (CPU) at 2.4GHz with 16 cores and an NVIDIA A100 GPU. O-RAN specifies an Acceleration Abstraction Layer (AAL) in between HAs and network functions such as DUs [44]. The AAL abstracts the O-Cloud resources as *Logical Processing Units (LPU)*. We implemented O-RAN's AAL using Intel Data Plane Development Kit (DPDK)'s Baseband Device Library (BBdev)[1]. Like DPDK's solutions for Ethernet, BBdev provides an abstraction for DU tasks through *devices* that can be used as O-RAN LPUs [91]. We implemented two LPUs to execute (*i*) the GPU DU processor, and (*ii*) instances of Intel FlexRAN [87] on a CPU pool for fallback DU operation, which is triggered by an LPU broker on top of the LPUs. YinYangRAN controls the O-Cloud by enforcing actions a_t using O-RAN O2 interface, and receives DU load data ϕ_t through O1 interface, as shown in Figure 5.6. According to O-RAN, the O-Cloud infrastructure is locally managed by an Infrastructure Management Service (IMS). To this end, we implement an interface between the GPU driver and the IMS, to control the allocation of SM resources, and another interface between the LPU Broker and the IMS, to fall back to DU software processing during the GPU re-configuration period. RUs and UEs are simulated following real-world wireless load patterns like those presented in Section 3.1. To this end, we encode and modulate the corresponding user data according to the 5G specification and add noise to match the observed patterns. We also let the same ML model we introduced in Subsection 5.1.4 to concurrently use the spare GPU resources of the O-Cloud.

[1] https://doc.dpdk.org/guides/prog_guide/bbdev.html

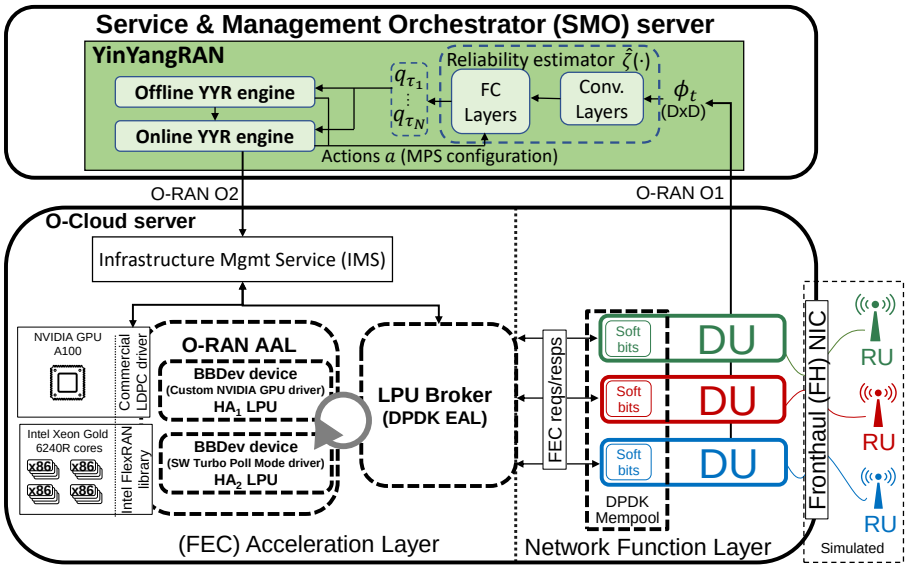


Figure 5.6: YYR system implementation.

Note that by running concurrently the ML model with the DU in the same GPU, the system ensures 2x cost gains with respect to the approach based on dedicating individual GPUs of the O-Cloud to individual services.

We implemented YinYangRAN using Python and PyTorch. For the reliability estimator $\hat{\zeta}(\cdot)$, we use a neural network with two convolutional layers of 8 and 4 filters of dimension 3x3, respectively, and two fully connected layers of 256 units each. The convolutional layers receive as input the traffic characterization ϕ and the fully connected layers receive the output of the convolutional layers plus the selected action a_t. We set the size of the traffic characterization histogram to $D = 5$ and we learn $N = 10$ quantiles. In our evaluations, we consider the lowest quantile $\tau_1 = 0.005$ to be higher than the reliability target, i.e., $\hat{q}_{\tau_1} > 1 - \epsilon$. We train $\hat{\zeta}(\cdot)$ offline considering diverse traffic loads and SNR patterns obtained from real-world traces introduced in Section 3.1.

5.6. Experimental evaluation

We compare both of our solutions, YYR-Full and YYR-Lite, with a baseline approach that deploys both DU and ML workloads on the same GPU using conventional software concurrency methods, i.e., with no SM isolation between them (see Subsection 2.5.5). Unless otherwise stated, we set $\epsilon = 0.1$, the decision period duration $\delta = 1$ s, we conservatively set $r = 0.5$ s, and present mean performance values with standard errors as error bars.

5.6.1. Impact of the reliability target

We first assess the system performance for different reliability targets. To this end, Fig. 5.7 shows the relative DU reliability and the ML throughput for different targets ϵ as shown in the x-axis. We observe both YYR-Full and YYR-Lite adapt to ϵ, trading off ML throughput as ϵ gets smaller (more stringent target). In contrast, a DU-agnostic baseline solution achieves poor reliability performance as it relies on the default GPU scheduler, which aims to be fair between workloads, which penalizes the inelastic DU demands.

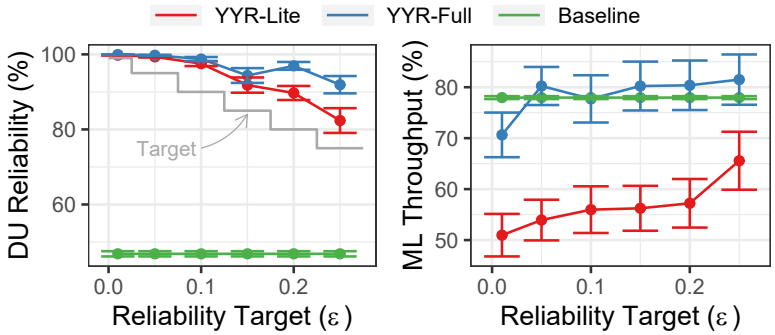


Figure 5.7: DU reliability (left) and ML throughput (right) *vs* reliability target ϵ. Comparison between YYR-Full, YYR-Lite, and a baseline.

Note that YYR-Full achieves higher ML throughput than YYR-Lite. This is because, though the re-configuration overhead is substantial ($r/\delta = 0.5$), YYR-Full takes into account the joint impact of re-configuring on the instantaneous cost and on the future cost (farsighted decision-making). In contrast, YYR-Lite selects the best action every decision period ignoring the impact of the re-configuration overhead in both the short and long terms (myopic decision-making).

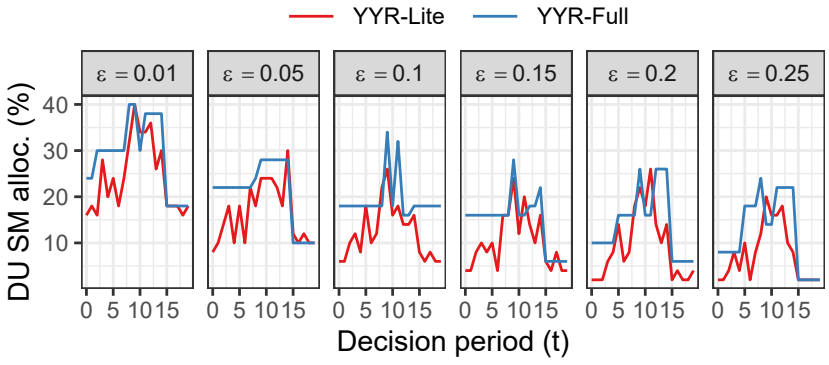


Figure 5.8: Evolution of the DU SM allocation for a subtrace of 20 decision periods.

We can confirm this in Figure 5.8, which shows the temporal evolution of the DU SM allocations made by our solutions during 20 decision periods of the above experiments.

Both YYR-Full and YYR-Lite adapt the allocation of SM resources to the time-varying load, which explains the ML throughput variance shown in Figure 5.7 (right), to guarantee meeting the reliability target. However, the myopic nature of YYR-Lite enforces a substantially higher number of re-configurations than YYR-Full, which explains YYR-Lite's throughput loss.

5.6.2. Impact of the traffic estimation error

Although the farsighted decision-making of YYR-Full shows better performance than YYR-Lite, it relies on traffic predictions $\tilde{\phi}$ that can deviate from the actual traffic observed in the network. To evaluate this, Figure 5.9 depicts the ratio of the cost of YYR-Full over YYR-Lite when we artificially induce different traffic estimation errors. To this end, we add a zero-mean Gaussian error with variance σ^2 to the output of our estimator. The figure shows values of σ relative to the mean load observed. When the traffic estimations are very accurate ($\sigma \to 0$), we observe around 20% cost savings when using YYR-Full with respect to using YYR-Lite. Evidently, these savings reduce as σ increases. However, even when σ is equal to the mean load estimated ($\sigma = 100\%$), YYR-Full achieves around 10% cost savings in average over YYR-Lite.

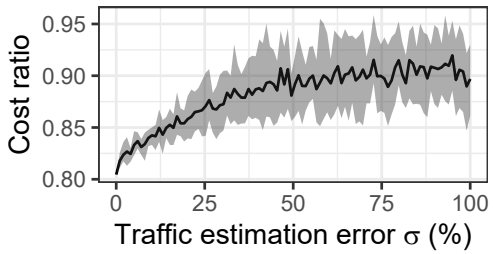


Figure 5.9: Ratio of the cost (eq. (5.2)) of YYR-Full over YYR-Lite as a function of the prediction error. The shaded area cover the 10th and 90th quantiles of the measurements.

5.6.3. Decision period

In the experiments shown before, YYR-Full shows better performance than YYR-Lite because the latter is a myopic approach that does not consider the cost of frequently re-configuring the GPU. As we discussed in Section 5.4, this cost depends on the ratio between re-configuration time r and the duration of each decision period δ. To assess this, we show in Figure 5.10 the cost and the ratio of re-configuration (percentage of time periods that the algorithms re-configures the GPU) for two decision lengths δ: one second (as in the previous experiments), which sets $r/\delta = 1/2$ and one minute, which sets $r/\delta = 1/120$. When we $\delta = 1$ min, YYR-Lite provides the same performance as YYR-Full because the impact of greedily re-configuring the GPU every decision period is negligible.

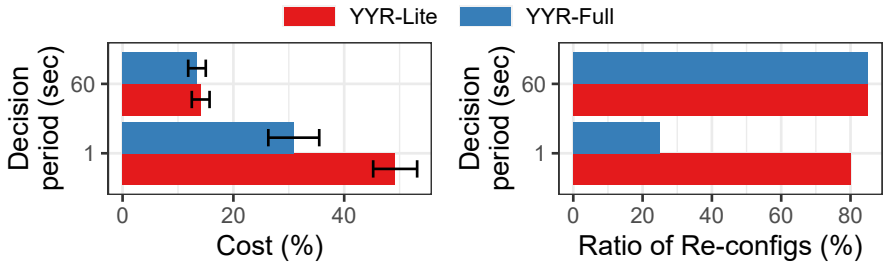


Figure 5.10: Decision period impact on cost and re-configurations for both YYR versions.

5.6.4. GPU Capacity

Finally, we evaluate the impact of the GPU capacity (in terms of the number of available SMs) on the system performance for all the solutions. Figure 5.11 depicts the DU reliability and the ML throughput as a function of the available number of SMs. YYR-Full and YYR-Lite attains the reliability target ($\epsilon = 0.01$) in all cases at the cost of ML throughput losses. The baseline solution also sacrifices ML throughput when the GPU capacity shrinks, but it sacrifices DU reliability as well without any prioritization.

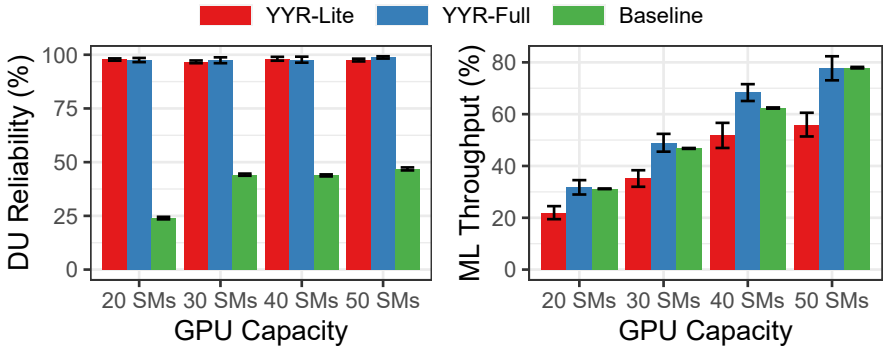


Figure 5.11: Impact of the GPU capacity on DU reliability and ML throughput.

6 Energy and cost efficiency in vRANs with Real-Time control

The solutions presented in Chapter 4 and Chapter 5 rely on control algorithms that respectively operate in the Near-RT-RIC (timescale of $\sim 10-100$ ms) and in the Non-Real-Time-RIC (timescale of ≥ 1 s). In particular, ECORAN depends on the assumption that traffic distribution changes slower than the time granularity of the Near-RT-RIC decision period, and YinYangRAN on the assumption that traffic is predictable throughout the duration of the Non-Real-Time-RIC decision interval. Therefore, to ensure reliability in the virtualized Radio Access Network (vRAN), both solutions adopt a conservative approach by configuring resources for the highest expected peak over their entire decisional timeframe, that leads to wasting resources due to overprovisioning.

However, based on insights from tracking workload dynamics in real-world cells, this chapter demonstrates that Distributed Unit (DU) traffic presents significant burstiness at Transmission Time Interval (TTI) level (timescale of 1 ms). Thus, additional gains in energy and cost efficiency can potentially be achieved by exploiting a heterogeneous O-Cloud infrastructure with both Hardware Accelerators (HA) and Central Processing Units (CPU) through a real-time controller capable of responding to TTI-level traffic fluctuations. To address this, the current chapter introduces CloudRIC, a real-time brokering system powered by lightweight data-driven models that jointly coordinates a centralized access for multiple DUs to a shared heterogeneous pool of computing processors, including HAs and CPUs, and assists centralized DUs with compute-aware radio policies while meeting vRAN-specific reliability targets.

6.1. Real-world traffic fluctuations

The traces introduced in Section 3.1 present remarkable burstiness at TTI level: Figure 6.1 shows a 10-second cell traffic snapshot that reveals wide fluctuations within milliseconds. This observation suggests that, to harness pooling opportunities in real-world cells while meeting the 5-nines reliability target, it is crucial to develop real-time control schemes that can react appropiately upon quick yet infrequent load peaks.

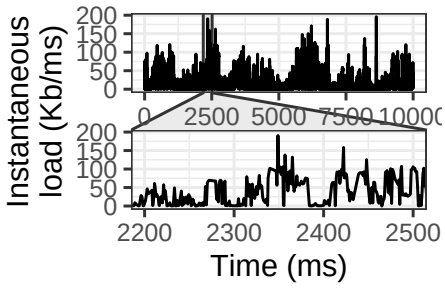


Figure 6.1: Instantaneous load fluctuations in a cell.

6.2. O-RAN AAL policies

The insights of the previous section combined with the analysis of Chapter 3 promote a real-time sharing of both high-performance HAs and low-consuming CPUs across multiple DUs. To inform a solution design for a cost-effective yet reliable vRAN, we next explore the space of possible policies balancing DU workloads among shared heterogeneous processors at sub-ms timescales.

The experiments involve a varying number of 100-MHz 5G DUs, with 5 active "x4" User Equipment (UE) each, that aggregate workloads between 85 Mb/s and 5 Gb/s. We also set the latency deadline $D = 3$ ms (for the moment), discard overdue Transport Block (TB)s and consider CPU and Graphics Processing Units (GPU)-based HA as processors. In this case, each DU locally subscribes to two Logical Processing Units (LPU)s, each handling one shared processor. As explained in Section 2.4, each DU maintains exclusive local access (with its own queues, as depicted in Figure 2.4). Then, for every TB, the corresponding DU selects an LPU following a *policy*.

With this setup, we study conceptual policies of increasing complexity, several of which are not viable in practice as they assume perfect knowledge of the future or functions not supported by the current O-RAN standard architecture. Yet, our goal now is studying potential gains; we will present a practical solution from Section 6.3.

6.2.1. Coordination

The first policy, "O-Greedy", follows a simple heuristic: given a TB, the DU selects the faster HA, as long as its local LPU queue is not full; otherwise, it falls back to CPU. Figure 6.2 (left) shows the system reliability (i.e., the fraction of TBs processed within D) as a function of the aggregated network demand. Reliability quickly drops when the demand exceeds 200 Mb/s as the local LPU queues, specific to each DU, grow too much with respect to the capability of the HA to handle the demand of all DUs, which results in TBs frequently missing the deadline D. When all the LPU queues saturate, all TBs miss D and reliability drops to 0%.

Inspired by [109], in "O-MWT" (O-RAN Minimum Waiting Time (MWT)) DUs compute the completion time at each LPU *by looking at the occupancy of its LPU queues* for every TB, and then select the fastest one. We model the performance of both LPUs using the dataset in Section 3.3, and let DUs have perfect knowledge of the exact processing latency of each new TB *a priori* to make optimal LPU choices. Though not realistic in practice, this policy illustrates the potential gains from exploiting *queuing information*, which in Figure 6.2 amount to 18% higher reliability and 10% lower energy-per-bit cost on average. Yet, given sufficiently high workloads, "O-MWT" also drops to 0% reliability at loads of 2 Gb/s or higher.

Under "C-MWT" (Coordinated MWT), the DUs consider each other's LPU queues to calculate the actual completion time of new TBs based on the system-wide load. This approach further increases reliability by 15% and reduces the energy cost by 19% on average over O-MWT; yet, it requires shared knowledge of LPU states across DUs, which is currently not supported by O-RAN. Moreover, it still cannot avoid reliability loss.

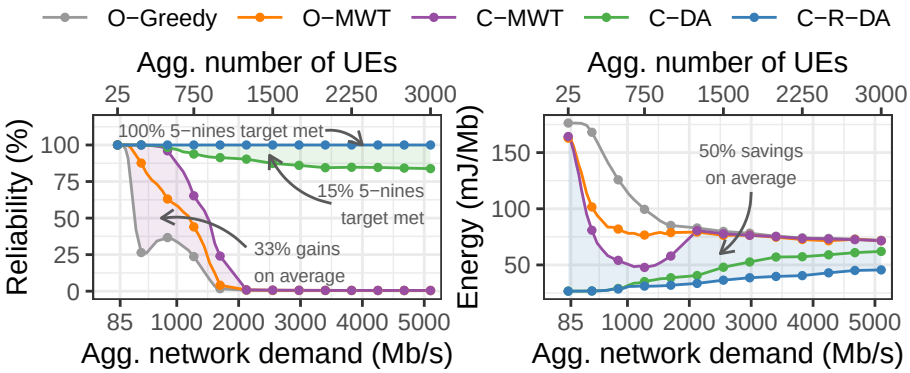


Figure 6.2: Performance of a heterogeneous O-Cloud with a varying number of DUs hosting "x4" UEs. Deadline $\mathbf{D} = \mathbf{3}$ ms.

6.2.2. Deadline awareness

Minimizing processing latency, as done by C-MWT, tends to overuse the HA, which incurs an energy toll, and neglect CPUs; moreover, when TBs are eventually allocated to the CPUs, it is done ignoring whether they can process those TBs within their deadline, which causes unreliability. Hence, we test a policy "C-DA" (Coordinated Deadline-Aware) that prioritizes CPUs. Under C-DA, given a TB, the DU calculates whether the TB can be processed within D by the CPU pool, and only offloads the TB to the HA otherwise. Thus, in addition to coordination, the policy requires *deadline awareness*. Figure 6.2 shows that C-DA boosts reliability and enables *opportunistic offloading* to high-performance HAs that save up to 80% of the energy cost for low-loaded scenarios.

6.2.3. Joint radio and computing control

All previous policies fall short of achieving the 5-nines reliability target, even under mild average workloads. This is due to the burstiness inherent to real workloads, discussed in Section 3.1, which occasionally introduces peaks that exceed the computing capacity.

The problem cannot be solved via LPU queue dimensioning: as the decoding latency of a single TB does not depend solely on its size but also on the associated Signal-to-Noise Ratio (SNR) and Modulation and Coding Scheme (MCS) (see Section 3.3), bounding the amount of bits allowed in a queue cannot deterministically bound the processing time and, hence, does not yield guarantees to meet deadlines. Instead, we argue that a better approach is to adapt the workload to the processing capacity *proactively*, through compute-aware radio policies that guarantee reliability.

We thus experiment with "C-R-DA" (Coordinated Radio-controlling Deadline-Aware), a policy that extends C-DA by throttling down radio grants when required to ensure that every TB can be processed in time afterwards. Note that, to explore the maximum gain of such a *compute-aware radio scheduling* strategy, C-R-DA has access to perfect knowledge of the future requests, which is clearly not possible in practice. However, C-R-DA shows how to efficiently trade off radio resources—*wasted* by the other policies anyway—for reliability when the O-Cloud gets congested and how it addresses the CPU reliability concern of Section 3.2. Indeed, Figure 6.2 shows that C-R-DA consistently meets the 5-nines reliability target with a 18% lower energy cost than C-DA on average.

6.3. System design

CloudRIC is an O-RAN compliant model that implements in practice the "C-R-DA" policy presented in the previous section with a three-fold goal, ordered by priority: (*i*) processing the DU workload within a predefined deadline D with 99.999% probability; (*ii*) maximizing network throughput; and (*iii*) minimizing energy consumption. A high-level view of a CloudRIC-powered O-Cloud is depicted in Figure 6.3. Our solutions integrates seamlessly into the standard O-RAN architecture, to which it adds two key enhancements, highlighted in red and in blue in Figure 6.3:

- A "Real-Time (RT) RAN Intelligent Controller (RIC) (RT-RIC)", which audits radio grants issued by DUs so as to guarantee that all the scheduled TB can be processed by the O-Cloud on time. Different from the existing O-RAN RICs, our RT-RIC assists DUs with compute-aware radio policies in real time.
- An "Acceleration Abstraction Layer (AAL) Broker (AAL-B)", which presents DUs with a single abstraction of the O-Cloud, and enables a centralized coordination of its processors, via two sub-components:
  - "AAL-B User Plane (UP)" (AAL-B-UP) acts as a proxy between O-RAN

Network Functions (NF) (DUs, in our case) and O-RAN AAL. From the NFs' viewpoint, the AAL-B-UP behaves as a virtual LPU that abstracts all the resources in the O-Cloud. From the perspective of the AAL, the AAL-B-UP appears as a virtual DU that is associated with the actual LPUs. Its job is routing arriving TBs from UEs to an LPU that is assigned by the AAL-B-CP.

- "AAL-B Control Plane (CP)" (AAL-B-CP) schedules granted TBs to LPUs in a way to guarantee that processing deadlines are met at minimum energy cost.

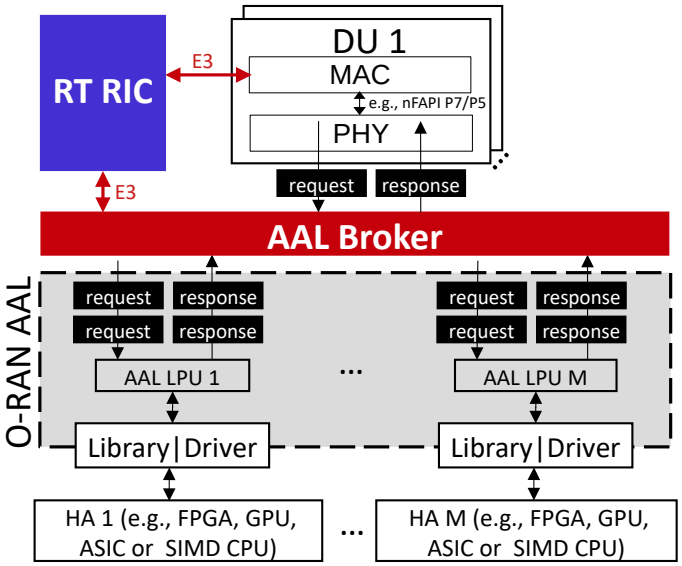


Figure 6.3: CloudRIC powered O-Cloud high-level architecture.

Given the real-time nature of the system, we expect DUs, RT-RIC and AAL-B to be deployed in the same physical infrastructure (e.g., an edge data center), and to be connected via a *new* E3 real-time interface that may be implemented using shared memory or low-latency tools such as Zenoh [110]. Our approach achieves three key results:

1. By centralizing the allocation of computing resources with AAL-B, we can efficiently police the load from multiple DUs across heterogeneous processors, as per Subsection 6.2.1;
2. By controlling the DUs grants, the RT-RIC can throttle radio resources when LPU queues are congested and ensure that deadlines are met, as per Subsection 6.2.2 and Subsection 6.2.3;
3. By decoupling AAL-B-CP and AAL-B-UP, we can minimize user plane overhead as it will be shown later (see Section 6.3.1 and Section 6.4.1).

6.3.1. Detailed design and workflow

Figure 6.4 details the design of CloudRIC for uplink operation, the most challenging in vRANs. The workflow follows steps ①–⑥, whose time budget is shown in Table 6.1.

Component	Operations	Time budget (μs)	Validation
RT RIC	① - ④	TTI $\in \{250, 500, 1000\}$	§6.4.2
AAL-B-CP	⑤	~200	§6.4.2
AAL-B-UP	⑥	D (1000 - 3000, see §2.1)	§6.4.1, §6.5

Table 6.1: Time budget for CloudRIC operations.

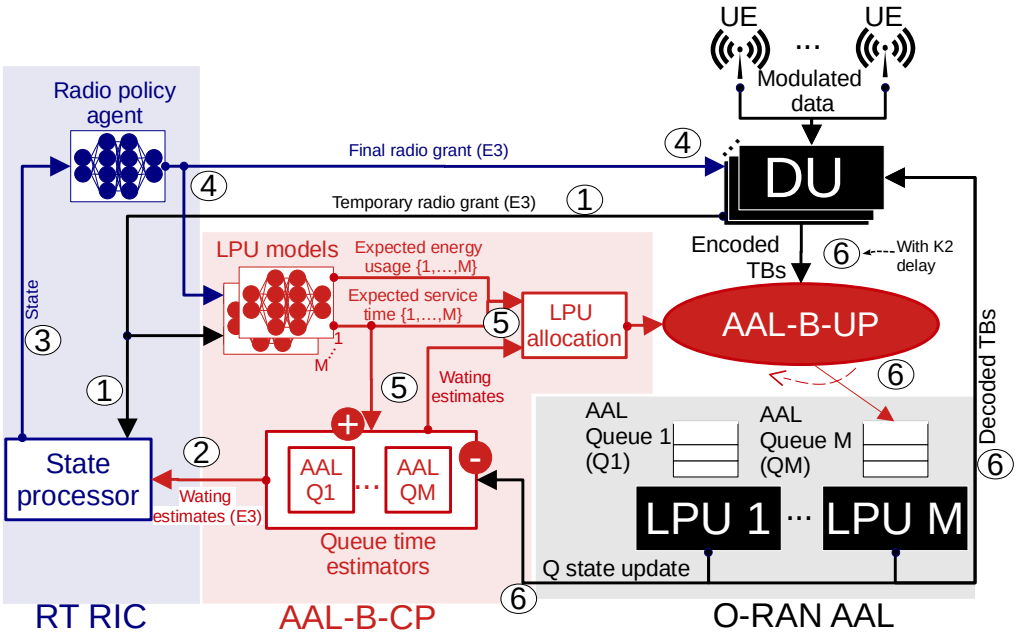


Figure 6.4: Detailed architecture of CloudRIC.

① Temporary grants. Each radio grant $\bar{g}^{(i)}, i \in \{1, 2, \dots\}$ generated by the DUs becomes a *temporary grant*. Prior to being allocated, it is sent to the RT-RIC, as a tuple containing the bandwidth $\bar{b}^{(i)}$ (number of Resource Block (RB)), selected MCS $m^{(i)}$, UE's SNR $s^{(i)}$, and corresponding TB size (bits) $\bar{t}^{(i)}$, i.e.,

$$\bar{g}^{(i)} := [s^{(i)}, m^{(i)}, \bar{b}^{(i)}, \bar{t}^{(i)}].$$

We use the top bar $\bar{\cdot}$ to indicate that element $\cdot$ is *temporary*.

② Latency estimation. The RT-RIC gets from AAL-B-CP estimates of the waiting time $\hat{w}^{(i)} := [\hat{w}_1^{(i)}, \dots, \hat{w}_M^{(i)}]$ at each LPU queue $Q_n^{(i)}$, for all LPUs $n \in \mathcal{L} := \{L_1, \dots, L_M\}$ exposed by O-RAN AAL. The top hat $\hat{\cdot}$ indicates that $\cdot$ is *a prediction*. *Queue time estimators* in AAL-B-CP produce $\hat{w}_n^{(i)}$ by keeping track of the time foreseen to serve all grants already queued in each $Q_n^{(i)}$, $\forall n \in \mathcal{L}$. These estimators perform simple calculations that rely on the (expected) LPU processing time $\hat{d}_n^{(k)}$ of each grant $g^{(k)}$ in $Q_n^{(i)}$, and on (real) time elapsed since the head-of-line grant started to be served. Step ⑤ explains how we derive $\hat{d}_n^{(i)}, \forall i, n$.

③ Radio policy. The RT-RIC *state processor* consolidates all the above information into a single feature vector $x^{(i)}$,

$$x^{(i)} := [s^{(i)}, m^{(i)}, \bar{t}^{(i)}, \hat{w}^{(i)}].$$

Given $x^{(i)}$, a *radio policy agent* ρ computes a radio allocation policy that is expressed as $r^{(i)} := \rho(x^{(i)}) \in \mathbb{R}_{0 \leq r^{(i)} \leq 1}$, where $b^{(i)} := r^{(i)} \cdot \bar{b}^{(i)}$ is the bandwidth (RB count) allowed for the final grant ($b^{(i)} \leq \bar{b}^{(i)}$). The design of ρ is in Section 6.3.2.

④ Final grant. Policy $r^{(i)}$ determines a final grant

$$g^{(i)} := [s^{(i)}, m^{(i)}, b^{(i)}, t^{(i)}]$$

for the UE, where its TB size $t^{(i)}$ is computed as per Third Generation Partnership Project (3GPP) specs [92]. Importantly, the grant must be compiled within the granularity of a TTI, as shown in Section 2.1 and in Table 6.1 for the time budget of operations ①–④. Then, as shown in Figure 6.4, $g^{(i)}$ is communicated to both the corresponding DU and the AAL-B-CP through interface E3. The DU can now notify $g^{(i)}$ to the UE as specified by 3GPP.

⑤ LPU allocation. The AAL-B-CP also receives $g^{(i)}$, which is used to allocate the computing resources necessary to process the grant. To this end, *LPU models* $\{\nu_n, \mu_n\}$ of each LPU $n \in \mathcal{L}$ are used to estimate the latency $\nu_n(g^{(i)}) := \hat{d}_n^{(i)}$, without queuing, and energy $\mu_n(g^{(i)}) := \hat{e}_n^{(i)}$ required by LPU n to process the TB associated with grant $g^{(i)}$. As they are stateless models, $\{\nu_n, \mu_n\}$ can be built with neural networks trained offline for each LPU. We present their design in Section 6.3.3.

Informed by the output of all LPU models ($\hat{d}_n^{(i)}$, $\hat{e}_n^{(i)}$) and queue time estimators ($\hat{w}_n^{(i)}$ from ②), the *LPU allocation function* uses the greedy Algorithm 2 to pre-assign one LPU to TB $g^{(i)}$. Algorithm 2 first prunes those LPUs that do not have queuing room, which yields $\mathcal{L}_1 \subseteq \mathcal{L}$. Then, using the waiting $\hat{w}_n^{(i)}$ and processing time estimates $\hat{d}_n^{(i)}$, it removes from $\mathcal{L}_1$ all LPUs for which $\hat{w}_n^{(i)} + \hat{d}_n^{(i)}$ falls outside a guard period δ of the deadline D, which yields $\mathcal{L}_2 \subseteq \mathcal{L}_1$. Finally, the LPU $k \in \mathcal{L}_2$ that can process the TB with the least amount of expected energy $\hat{e}_k^{(i)}$ is selected. The result is communicated to the AAL-B-UP (to route the grant) and to the relevant queue time estimator (to update its estimate).

As explained in Section 2.1, UEs must send the granted TBs within $K2$ slots, which sets a worst-case deadline of $\sim$200 μs for this step (numerology $\mu = 2$ and $K2 = 0$). While simple, Algorithm 2 proves both efficient and extremely fast.

⑥ LPU processing. After $K2$ slots, the UE transmits the granted TB $g^{(i)}$. Once received, the DU forwards the TB to the AAL-B-UP's dispatcher, which routes the TB to the pre-assigned LPU queue for processing. Once decoded, the TB data is sent to the DU and the corresponding queue time estimator in the AAL-B-CP is updated. As explained in Section 2.1, this step must be completed within D.

Algorithm 2: Greedy LPU allocation

Input: $D, t^{(i)}, \hat{w}_n^{(i)}, \hat{d}_n^{(i)}, \hat{e}_n^{(i)}, \forall n \in \mathcal{L}$
$\mathcal{L}_1 = \mathcal{L}_2 = \emptyset$
for $n \in \mathcal{L}$ **do**
  if *Available Space in* $Q_n > t^{(i)}$ **then**
    $\mathcal{L}_1 \leftarrow \mathcal{L}_1 \cup n$
for $n \in \mathcal{L}_1$ **do**
  if $\hat{w}_n^{(i)} + \hat{d}_n^{(i)} < D - \delta$ **then**
    $\mathcal{L}_2 \leftarrow \mathcal{L}_2 \cup n$
if $\mathcal{L}_2 \neq \emptyset$ **then**
  $L \leftarrow \arg\min_{k \in \mathcal{L}_2} \hat{e}_k^{(i)}$
else
  $L \leftarrow \arg\min_{k \in \mathcal{L}_2} \left(\hat{w}_k^{(i)} + \hat{d}_k^{(i)} \right)$
Output: L

6.3.2. Radio policy agent

CloudRIC's radio policy agent shall produce grants that are conscious of the current pressure on the LPUs and of the added load brought by the new requests. In turn, these properties depend on the specific hardware and software settings of the network environment (e.g., the number, type, and detailed specifications of the LPUs, which substantially affect the performance as we saw in Section 3.3), and are stateful (i.e., are contingent on the utilization of LPUs at the moment of the radio scheduling decision). Hence, they are hard to model in advance, and require a solution that automatically adapts itself to the target O-RAN system upon deployment.

In light of this consideration, we resort to a data-driven model that can learn the tangled relationships above at runtime. We implement the radio policy agent as a *soft actor-critic* deep Reinforcement Learning (RL) algorithm, under a twofold rationale: (*i*) the RL paradigm allows training the model online from system observations directly; and (*ii*) inference is achieved with a simple neural network (the actor) that involves trivial arithmetic operations to minimize inference latency, as later proven in Section 6.4. Among many actor-critic instances, we opted for a soft version that maximizes the reward while acting as randomly as possible to explore the solution space during training; this model achieves state-of-the-art performance in tasks similar to the one we tackle [111].

The instantaneous reward function used by the critic is

$$R^{(i)} = \begin{cases} t^{(i)}/\bar{t}^{(i)} & \text{if } w_n^{(i)} + d_n^{(i)} < D - \delta \\ -\beta & \text{otherwise.} \end{cases} \tag{6.1}$$

Two cases are possible. If TB $g^{(i)}$ is processed within a guard period δ to the deadline D, the decision is assigned a positive reward proportional to the granted fraction of the requested TB ($t^{(i)}/\bar{t}^{(i)}$). Otherwise, penalty β is assigne (negative reward).

6.3.3. LPU models

The purpose of the LPU models $\{\nu_n, \mu_n\}$ is to estimate the processing latency, without queuing delays, and the energy consumption of each processor in the pool, for a given grant $g^{(i)}$. This is a stateless task, hence the LPU models can be easily built offline with data collected in a pre-calibration phase. Indeed, we take this approach in our implementation of CloudRIC, employing the measurement data presented in Section 3.3 to train LPU models for the specific Low-Density Parity-Check (LDPC) drivers and GPU/CPU LPUs considered there.

We realize the LPU models as simple feed-forward neural networks, which are accurate and efficient universal function approximators. An important remark concerns the loss function used to train the models. For the energy estimator μ_n, we use a legacy Mean Absolute Error (MAE) loss to minimize the absolute error. However, the loss for the processing latency estimator ν_n has a dedicated design that stems from the following consideration. Underestimating $\hat{d}_n^{(i)}$ risks violating deadlines, which incurs significant throughput loss and poor spectrum usage; conversely, overestimating $\hat{d}_n^{(i)}$ only implies far less disruptive radio resource under-allocations. Therefore, it is critical that the LPU model never underestimates latency, while still trying to minimize overestimation. Hence, we train ν_n with an asymmetric loss function that we call Worst-Case Estimation Time (WCET) loss. Let the latency prediction error for grant $g^{(i)}$ be $\epsilon_n^{(i)} := \hat{d}_n^{(i)} - d_n^{(i)}$, $\forall n \in \mathcal{L}$; the WCET loss is then expressed as

$$L(\epsilon_n^{(i)}) = \begin{cases} -\lambda \cdot \epsilon_n^{(i)} & \text{if } \epsilon_n^{(i)} \leq 0 \\ \epsilon_n^{(i)} & \text{if } \epsilon_n^{(i)} > 0, \end{cases} \tag{6.2}$$

where $\lambda > 1$ allows avoiding optimistic predictions.

6.4. System implementation

Figure 6.5 illustrates CloudRIC implementation in a multi-DU server, on top of the O-RAN AAL introduced in Section 2.4 with heterogeneous GPU and Single Instruction Multiple Data (SIMD)-capable 16-core CPU processors. As mentioned in Section 3.4, a server with a conventional Peripheral Component Interconnect Express (PCIe) v3.0+ bus and a 100-GbE fronthaul interface can aggregate up to 3.8 GHz of spectrum, supporting dense small-cell networks [95]. Radio Unit (RU) and UEs are simulated following the bursty demand profiles presented in Section 3.1, which generate realistic workload fluctuations with varied levels of stress on our O-Cloud platform. To this end, we encode, modulate, add noise and demodulate signals locally.

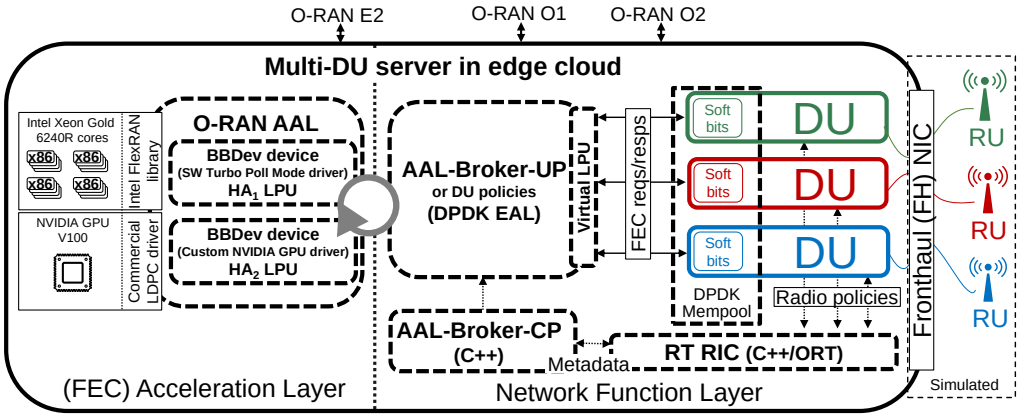


Figure 6.5: A CloudRIC-powered O-Cloud prototype.

6.4.1. User Plane: AAL Broker

The implementation of CloudRIC's AAL-B-UP (∼2K lines of C++) follows Figure 6.4, and builds on top of the O-RAN AAL we developed in Section 6.2. We use the Data Plane Development Kit (DPDK) Environment Abstraction Layer (EAL) to abstract the complexity of thread and memory management of four main threads.

The first thread runs the AAL-B-UP, as well as its interfaces with the AAL-B-CP and the DUs using shared memory to reduce latency overhead. Upon each TB arrival, the AAL-B-UP routes its data to an LPU queue pre-assigned by the AAL-B-CP, as shown in Figure 6.4. The second and third threads execute the CPU and GPU LPUs, respectively: they (*i*) gather bursts of TBs from the associated LPU queue; (*ii*) enqueue bursts in the LPU; (*iii*) execute the corresponding driver or library; and (*iv*) enqueue the decoded bits into an output queue. The last thread handles performance metrics.

Validation. To validate our implementation, we use all the profiles presented in Section 3.1 with a random LPU allocation, and measure the latency overhead introduced by our AAL-B-UP operations. Figure 6.6 (blue line) shows the distribution of the overheads, with a median latency of 2 μs and a 99th percentile of 46 μs, which is negligible compared to the 1-3 ms time budget available for the LPU processing of each TB according to the summary in Table 6.1.

6.4.2. Control Plane: AAL-B-CP and RT-RIC

The AAL-B-CP and the RT-RIC are implemented in C++ (∼1.5K lines) using DPDK for efficient inter-process communication, and Open Neural Network Exchange (ONNX) Runtime (ORT) for neural network acceleration [112]. The AAL-B-CP uses four threads implementing, respectively, the main pipeline of Section 6.3.1, the management of queue state updates received from the AAL, the queue time estimator logic, and the LPU models. The RT-RIC uses a single thread to implement the radio policy agent.

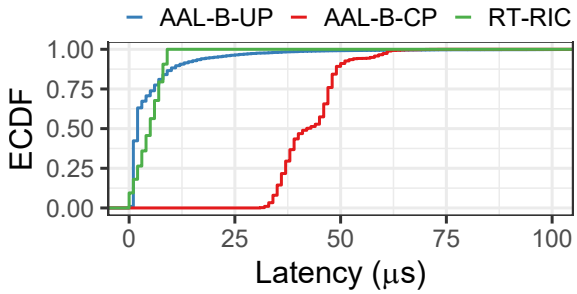


Figure 6.6: Overheads of RT-RIC and AAL-B models.

The policy selects an action $r^{(i)} \in \{0, 0.1, \ldots, 0.9, 1.0\}$ for every TB i. Both actor and critic are neural networks with (5x140, 140x140, 140x11) and (16x140, 140x140, 140x11) layers, respectively, that are jointly trained using equation (6.1) as reward, parameterized with $\delta = 10\%$ of the deadline D and $\beta = 10$, which experimentally proved to work well. For the target soft Q-function of the critic [111], we set a discount factor equal to 0.99.

The LPU models' neural networks are composed of three dense layers (3x128, 128x128, 128x1) and WCET loss in equation (6.2) is parametrized with $\lambda = 10$. To illustrate the advantage of training the latency model ν_n with our WCET approach, we used the dataset presented in Section 3.3 to show the distribution of the model's relative errors $\epsilon_n^{(i)}$ with solid lines in Figure 6.7. As a benchmark, we also plot with dashed lines the relative error of an identical neural network trained to minimize the standard MAE. What is relevant to CloudRIC operations is the fraction of overly optimistic estimates ($\epsilon_n^{(i)} < 0$) that lead to overrating the O-Cloud capacity and to queuing TBs that will later violate their deadline. Only 1.5% of the errors are optimistic for our method in contrast to the 53% of a MAE loss. Note that percent errors are higher for GPU than CPU, because the GPU latencies $d_n^{(i)}$ are considerably smaller, hence harder to estimate: the absolute error in μs is however comparable for the two processors.

Validation. As explained in Section 6.3, the overhead of the RT-RIC, which runs the the radio policy agent, and the AAL-B-CP, which runs the LPU models, have latency constraints indicated in Table 6.1. To assess that timing requirements are fulfilled by our implementation, we run CloudRIC for all the UE demand profiles presented in Section 3.1 and measure the overhead introduced by both modules. The distribution of the resulting latencies is depicted in Figure 6.6. The RT-RIC's radio agent (green line) has a median and a 99th percentile latency of 5 μs and 9 μs, respectively. The LPU model neural networks incur instead into a median and a 99th percentile latency of 42 μs and 62 μs, respectively (red line). Both meet their budgets and validate CloudRIC's real-time operation.

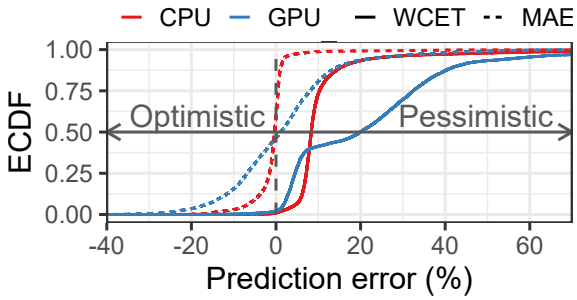


Figure 6.7: Prediction error of LPU models in AAL-B.

6.5. Experimental evaluation

We next perform a thorough experimental campaign. First, we quantify the training overhead of CloudRIC (Section 6.5.1). Next, we validate that CloudRIC meets the design goals set in Section 6.3 in terms of reliability, throughput, and energy (Section 6.5.2). Lastly, we juxtapose CloudRIC against several benchmarks, scrutinizing aspects of reliability, energy- and cost-efficiency (Section 6.5.3).

6.5.1. Training time

CloudRIC hinges on data-driven approaches to realize the LPU models and radio policy agent. As explained in Section 6.3.1, the LPU models are stateless and can be pre-trained, e.g., using the dataset introduced in Section 3.3 in our case. Instead, the radio agent is trained online upon deployment in the target system, and the training time it requires to learn effective radio policing decisions is an important metric.

To analyze its training cost, we initialize RT-RIC's radio agent with random weights and study the evolution of its performance as it learns to schedule the traffic of 150 UEs distributed among thirty 100-MHz DUs. Figure 6.8 presents the mean normalized reward achieved by CloudRIC after it has observed an increasing number of TBs across three processing deadlines $D = \{1, 2, 3\}$ ms commonly used in the literature [12, 13]. We present results for "x4", "x32", and "Max" UEs, as introduced in Section 3.1, to assess low, medium, and high workloads. CloudRIC reaches maximum reward almost instantaneously for small workloads ("x4"): the radio agent readily learns that it does not need to constrain the allocation of radio resources to guarantee deadlines in this case. With more demanding "x32" and "Max" users, CloudRIC takes up to 50K TBs to converge, and achieves 90% of that reward with 10 times less TBs. It shall be noted that 50K TBs correspond to less than 4 s in real-time with "x32" users, which demonstrates that the radio agent can learn good policies with minimal service degradation upon deployment.

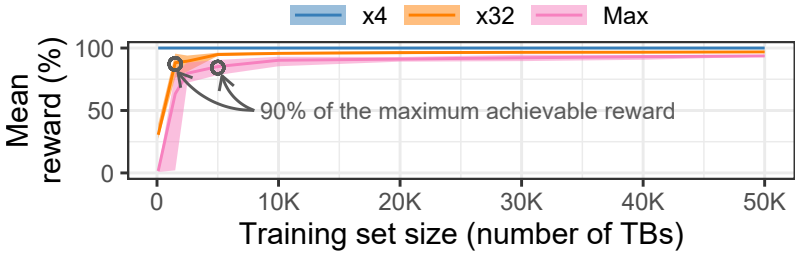


Figure 6.8: CloudRIC's mean and max-min range reward for different training sets.

6.5.2. CloudRIC validation

As introduced at the beginning of Section 6.3, CloudRIC has three goals, prioritized as: (*i*) achieving 5-nines reliability, (*ii*) maximizing throughput, and (*iii*) minimizing energy use, which we evaluate next.

Reliability. Compliance with the processing deadline during the inference phase directly controls the reliability of CloudRIC. We now better substantiate the dependability of our solution for real-time operation after the (very quick, see Subsection 6.5.1) online training phase. Figure 6.9 shows the empirical distribution of the processing latency for all scenarios in Section 6.5.1. The shaded areas highlight deadline violations: by bounding all distributions to the left of those regions, CloudRIC meets the reliability target in all cases. More precisely, Table 6.2 depicts the latency's 99.999^{th} percentiles, showing that CloudRIC consistently grants 5-nines reliability, which validates the first design goal. To achieve this, with growing loads and (to a minor extent) with more stringent deadlines, CloudRIC increases the average use of both CPU and HA as shown Figure 6.10 (top).

However, when LPUs are under pressure, CloudRIC must bound the allocation of radio resources as depicted in Figure 6.10 (middle). That is, in response to workloads that increasingly exceed the O-Cloud's capacity, the radio policy agent throttles down a progressively higher fraction of radio resources that otherwise would be wasted. Despite this, CloudRIC attains on average 98.7% of the achievable throughput.

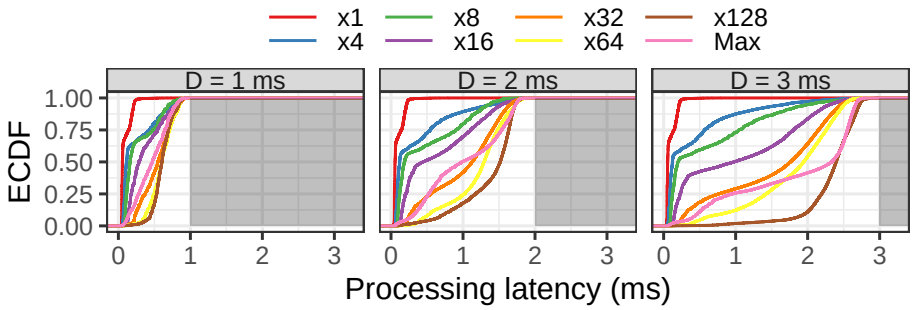


Figure 6.9: Processing latency distribution for different deadlines $\mathbf{D} = \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$ ms.

Deadline (D)	x1	x4	x8	x16	x32	x64	x128	Max
3 ms	0.79	2.84	2.99	2.81	2.82	2.84	2.99	2.79
2 ms	0.80	1.85	1.88	1.92	1.90	1.95	1.99	1.97
1 ms	0.73	0.99	0.96	0.99	0.96	0.97	0.97	0.98

Table 6.2: 99.999^{th} latency percentile (ms) for different deadlines D and demand profiles.

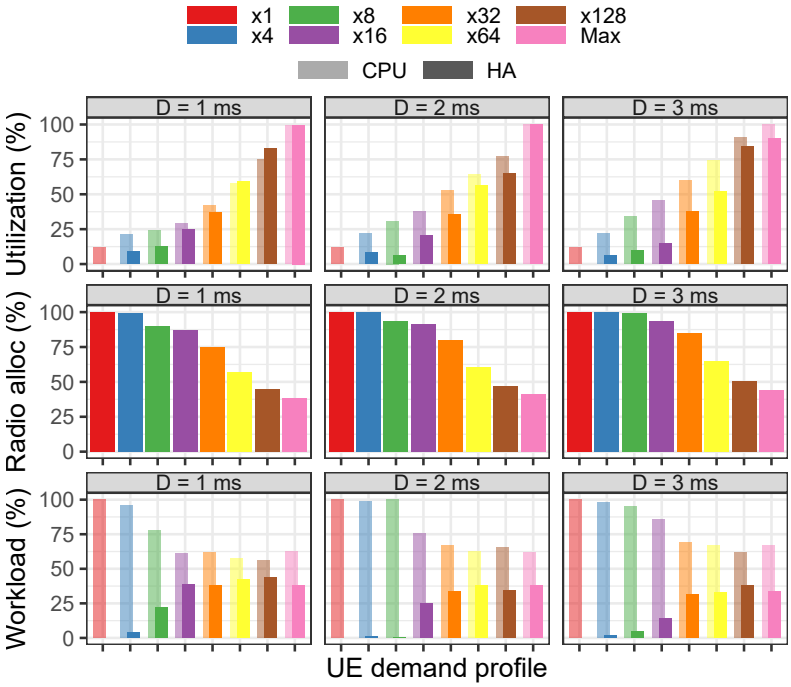


Figure 6.10: Mean resource utilization (top), radio allocation (middle), and workload allocation (bottom) for different deadlines $\mathbf{D} = \{\mathbf{1}, \mathbf{2}, \mathbf{3}\}$ ms and demand profiles.

It is worth noting that the results in Figure 6.10 above are averages, while the UE traffic at the ms timescale is inherently bursty (as exemplified in Figure 6.1). Therefore, CloudRIC must manage sudden upswings of requests (which saturate the LPUs and force radio grant limiting) alternating to low-demand periods (where all grants can be accommodated but resource utilization remains low). This explains why LPU usage inflation is fairly linear in the face of exponentially surging demands of "x1" through "Max" UEs, and why 100% radio allocations are not feasible even when the LPUs are not fully used, with "x8" through "x128" UEs.

Throughput and energy. We now focus on the last design goals, i.e., exploiting computing heterogeneity to maximize throughput and energy efficiency. To that end, CloudRIC's LPU allocation function, informed by the LPU models, prioritizes energy-prudent processors as long as they meet deadlines but resorts to more powerful HAs when strictly required to maximize throughput. This yields a flexible load balancing, as shown in Figure 6.10 (bottom): for instance, 100% of the workload is assigned to the

CPU pool to save energy in the case of "x1" UEs, but around 40% is allotted to the HA to protect reliability in higher-loaded scenarios with "x32" through "Max" UEs.

To study this, we deploy CloudRIC on different AAL platforms: a CPU-only, a HA-only, and a heterogeneous platform combining CPUs and HAs. To provide a fair comparison across platforms, we sized each platform (number of CPU cores, HAs, etc.) such that additional computing resources of any type could not improve performance any further. We then emulate five DUs on each platform and vary the load as we did in Section 3.2 for $D = \{1, 2, 3\}$ ms. In contrast to Section 3.2's results, Figure 6.11 proves that CloudRIC meets the 99.999% reliability target in all cases. More importantly, the throughput (relative to the load) and energy performance figures illustrate the advantages of a heterogeneous platform: there, CloudRIC matches the throughput of an HA-only platform at a similar energy cost to that of a CPU-only system. In contrast, CloudRIC must sacrifice 15-75% of throughput in a CPU-only platform to ensure reliability, and consumes 3-14× more energy per bit in an HA-only system. Indeed, the ability of CloudRIC to exploit CPUs opportunistically provides substantial gains in energy consumption. We analyze this in more detail in Figure 6.12. The figure shows the energy cost savings achieved by CloudRIC on a heterogeneous platform with respect to the cost of exclusively relying upon a HA. The results refer to CPU pools of varied size, under all UE profiles. Remarkably, CloudRIC provides over 50% energy cost savings with just a single CPU core in presence of low-load "x1" UEs. Increasing the CPU pool size provides more savings, but with diminishing returns as many cores remain unused. For higher workloads, the energy cost savings are understandably lower as the LPU allocation function prioritizes network throughput and reliability over energy consumption. For instance, a single-core CPU pool provides practically no gain for the case of "Max" users but a 16-core pool attains 37% savings for the same scenario.

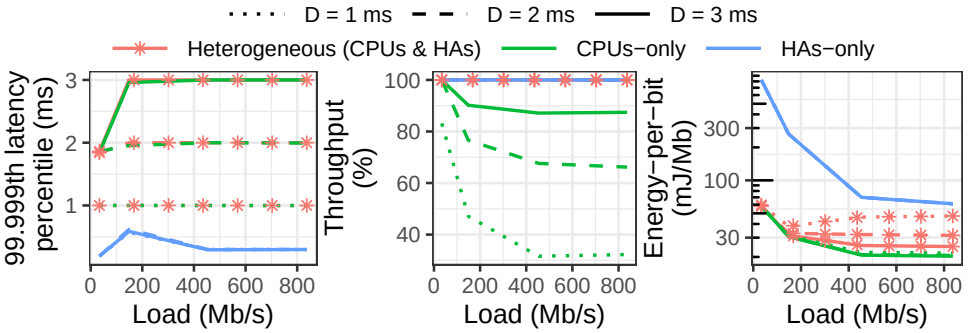


Figure 6.11: 99.999$^{\text{th}}$ latency percentile, throughput and energy consumption of CloudRIC on three different platforms.

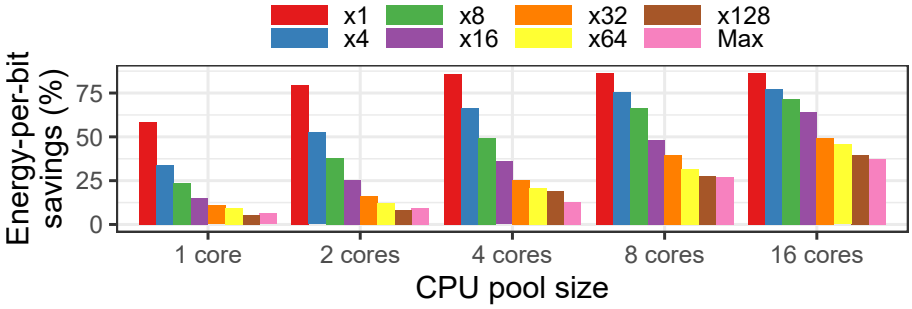


Figure 6.12: Energy savings of CloudRIC when using a CPU pool opportunistically relative to only using the HA.

6.5.3. Comparison with other approaches

CloudRIC improves reliability and sustainability of vRANs over both best practices that the operators currently adopt in their deployments, as well as advanced benchmarks that only partially implement our design guidelines.

Benchmarks. We show this by juxtaposing CloudRIC to the following baselines. "Industry-std" uses a dedicated *in-line* HA co-located with every DU, which is *the approach in the industry today.* To test this approach fairly, we simulate as many in-line HAs as DUs, each with the same performance of an NVIDIA V100 GPU, using the dataset of Section 3.3 and ignoring the look-aside overhead shown in blue in Figure 6.6.

The other benchmarks are the policies introduced in Section 6.2: "O-Greedy", "O-MWT", "C-MWT", and "C-DA", which are not simulated anymore but implemented in our shared and heterogeneous AAL platform, like CloudRIC. In Section 6.2, several of these approaches relied on future knowledge, which is not feasible for real-time operation. To enable real-time operation, we use LPU models trained with a MAE loss (as explained in Subsection 6.3.3) to predict the information they require.

We run comparative experiments with a varying number of 100-MHz DUs aggregating 50 to 350 UEs in a single server. Specifically, we associate 5 UEs to each DU and vary the number of DUs sharing the O-Cloud platform. Figure 6.13 depicts reliability (top), expressed as the percentage of TBs processed within D, energy efficiency (middle), calculated as the amount of data bits processed successfully per unit of energy), and cost efficiency (bottom), measured as the mean network throughput achieved per unit of capital investment. Note that, to calculate energy- and cost-efficiency, we ignore the baseline energy and financial cost of the servers, and use the representative market prices of Table 1.1, i.e., \$110 per CPU core, and \$3,000 per HA, making each HA-only "Industry-std" server 37% cheaper than the shared heterogeneous server. Results are for $D = \{1, 2, 3\}$ ms and a variable number of "x4" (left) and "Max" (right) UEs, hence DUs.

The first key remark is that only CloudRIC and "Industry-std" provide high reliability across all scenarios. Even "C-DA", which proves high reliability in case of "x4" users,

cannot meet 99.999% reliability given enough number of UEs and DUs (right-most side of the plot). Precise figures on this critical aspect for industry-grade vRANs are presented in Table 6.3: the 99.999$^{\text{th}}$ percentile of the processing latency confirms that "Industry-std" is the only benchmark to achieve the same 5-nine reliability of CloudRIC.

However, "Industry-std" matches the dependability of CloudRIC by largely over-dimensioning the computing capacity, which yields the worst cost-efficiency performance. Figure 6.13 demonstrates how CloudRIC makes the best use of a much smaller, shared, and heterogeneous O-Cloud to achieve 15x higher cost efficiency and 3x higher energy efficiency, on average, than the legacy industry practice.

The capital cost of all the other benchmarks is the same as CloudRIC's so its cost-efficiency gains are solely due to network throughput gains. Because "C-DA" uses CloudRIC's AAL-B, it achieves similar performance to CloudRIC for small workloads; however, the absence of the RT-RIC causes "C-DA" to waste radio and computing resources on TBs that miss their deadlines during heavier workloads, which results in 34% and 31% lower energy efficiency and throughput, respectively. "O-Greedy", "O-MWT", and "C-MWT" all confirm the poor reliability observed in Section 6.2 on a real-world platform, with throughput dropping to 0% for "Max" UEs.

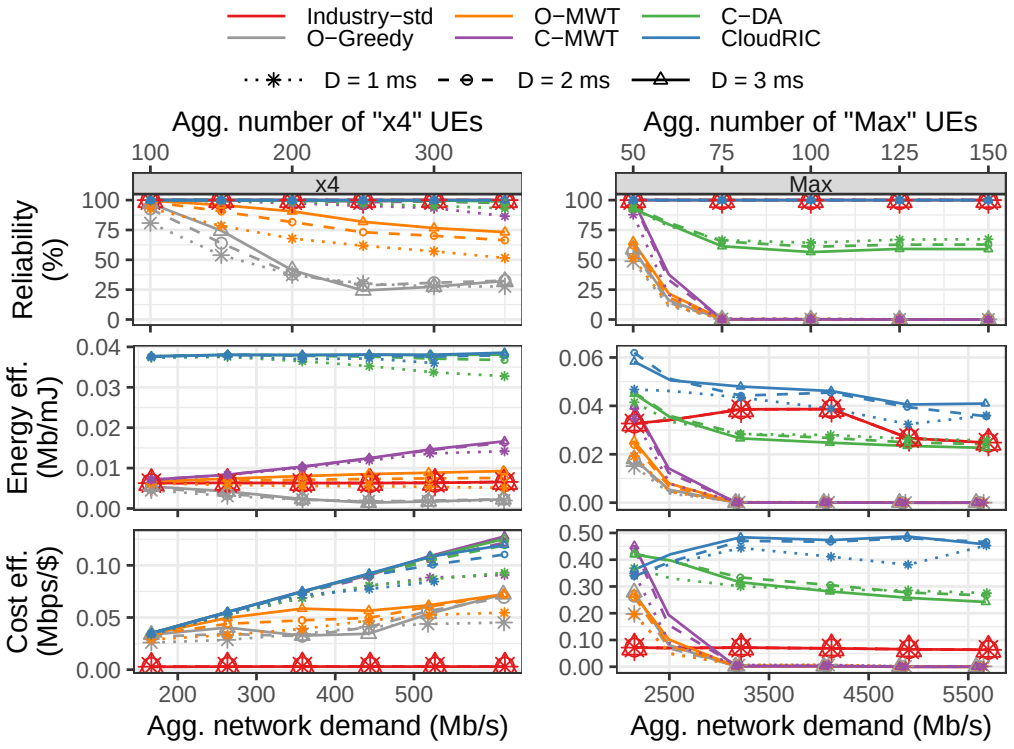


Figure 6.13: Reliability (top), energy-efficiency (middle), and cost-efficiency (bottom) for different policies on different scenarios with with "x4" (left) and "Max" (right) users.

UEs profile	Deadline (D)	CloudRIC	Industry-std	O-Greedy	O-MWT	C-MWT	C-DA
"x4"	3 ms	2.94	0.20	28.5	10.9	6.73	6.59
	2 ms	1.95	0.20	23.4	9.83	7.51	4.72
	1 ms	0.99	0.29	13.9	9.26	9.06	2.92
"Max"	3 ms	2.95	0.29	35.4	35.5	35.4	6.15
	2 ms	1.99	0.29	24.1	24.1	24.1	4.22
	1 ms	0.99	0.29	12.9	12.9	13.0	2.40

Table 6.3: Comparison of the 99.999$^{\text{th}}$ latency percentile (ms) of different approaches for the scenarios presented in Figure 6.13.

Different Hardware Accelerators. Finally, we analyze the impact of different HAs on the performance gains of CloudRIC over "Industry-std", i.e., the only two approaches that meet the reliability target of 99.999% at all times. To this end, we scale the latency and the power consumption measurements of the NVIDIA V100 GPU by factors $0.2 \leq \alpha_{\text{lat}} \leq 1$ and $0.2 \leq \alpha_{\text{pwr}} \leq 1$, respectively, to simulate diverse and future HAs with different combinations of α_{lat} and α_{pwr}. For instance, based on publicly available information, we roughly estimate that the look-aside Intel ACC100 HA may be modeled with $0.5 \lesssim \alpha_{\text{pwr}} \lesssim 0.7$, and the look-aside Intel ACC200 (integrated into the new Sapphire processors) or the in-line Qualcomm X100 HA with $0.6 \lesssim \alpha_{\text{lat}} \lesssim 0.8$.

To study different scenarios, we generate various workloads by emulating different numbers of DUs, each with five "x4" or "Max" UEs. Like before, one heterogeneous platform is shared among all the DUs in the case of CloudRIC, while we simulate a HA-only server per DU for "Industry-std". On the one hand, Figure 6.14 depicts the energy efficiency of CloudRIC relative to that of "Industry-std".

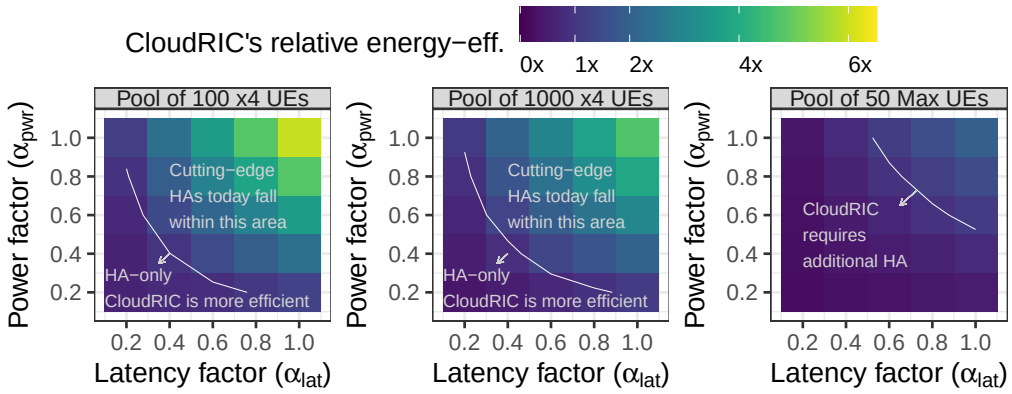


Figure 6.14: CloudRIC's energy-efficiency with respect to Industry-std's for different HAs characterized by scaling parameters α_{lat} and α_{pwr} for an NVIDIA GPU V100.

For small-medium workloads (aggregating <500 "x4" UEs), the heterogeneous platform enables CloudRIC to achieve substantial energy efficiency gains—up to 6× the efficiency of "Industry-std"—when the HA satisfies $\alpha_{\text{pwr}} + \alpha_{\text{lat}} \geq 1$. As mentioned above, cutting-edge HAs today are well within this range. When $\alpha_{\text{pwr}} + \alpha_{\text{lat}} < 1$, perhaps

achievable by future HA technology, the HA's latency and energy consumption is so low compared to today's CPUs that using the latter becomes inefficient. This occurs in a larger $(\alpha_{\text{pwr}}, \alpha_{\text{lat}})$ area for very large workloads (>750 "x4" or >10 "Max" UEs), where CloudRIC must throttle down radio resources to preserve reliability. Note that, in such extreme cases, CloudRIC may share more HAs to boost energy efficiency.

On the other hand, Figure 6.15 presents the cost efficiency of CloudRIC relative to that of "Industry-std". Since CloudRIC aggregates all the workload into a single server, its cost-efficiency gain over "Industry-std" grows linearly with the number of DUs. This is depicted in Figure 6.15 for different workloads, generated in the same way as before, and for different α_{lat} values (note that cost-efficiency is independent of α_{pwr} as it does not impact on throughput). The growth rate degrades when the computing resources available in CloudRIC's server are maxed out and CloudRIC has to sacrifice throughput to preserve reliability (e.g., >310 "x4" or >50 "Max" UEs). This is more severe for higher α_{lat} because reducing α_{lat} essentially increases the HA's computing capacity. Moreover, since cost-efficiency also depends on the number of DUs per server, high-loaded DUs with "Max" UEs incur lower gains as large workloads are generated with fewer DUs.

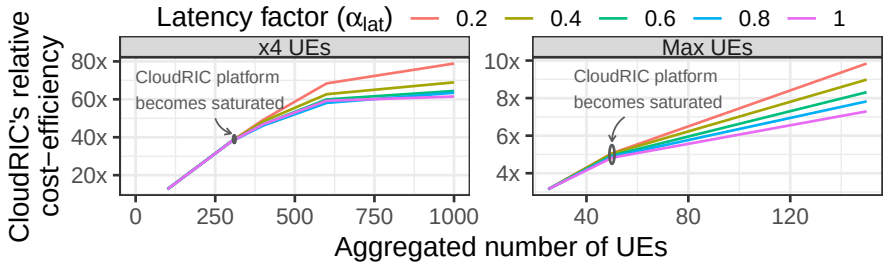


Figure 6.15: CloudRIC's cost-efficiency w.r.t. HA-dedicated approach with different α_{lat}.

7 Conclusions and perspectives

The growing need for Hardware Accelerators (HA) to support individual Distributed Unit (DU) of virtualized Radio Access Network (vRAN) has raised several concerns about the possibility of obtaining cost- and energy-efficient vRAN deployments that are capable of guaranteeing acceptable performance and reliability for the industry.

This thesis has demonstrated that it is possible to address these concerns by reducing the energy toll and amortizing the economic cost of the energy-intensive and expensive HAs while ensuring reliable operation of the vRAN. For this purpose, this thesis has proposed the following approaches: (*i*) the opportunistic usage of HAs supported by complementary energy-efficient software processors like Central Processing Units (CPU) to decrease the energetic costs, and (*ii*) centralizing the access of multiple DUs to a shared heterogeneous pool of computing processors including HAs to reduce deployment costs.

State-of-the-art approaches and prior works in literature have tried to address the aforementioned concerns, showing however substantial limitations in terms of coarse operational timescales (not able to react to the dynamics of real-world DU workloads) and in terms of sharing and multiplexing efforts with the available computing resources in the system, favoring rather dedicated and thus over-dimensioned approaches.

7.1. Conclusions

In this thesis, the shortcomings of previous works and state-of-the-art approaches have been addressed and important steps have been made towards the deployment of energy- and cost-effective vRANs, in which the latency of DUs processing stays below typical target deadlines with target probability values (reliable operation).

In the first part of this thesis, we have provided experimental foundation and justification for the design paradigms that are needed to deploy reliable vRANs while reducing costs. For the first time in literature, we have showed the latency and energy consumption performance of Graphics Processing Units (GPU)-based HA when executing one of the most compute-intensive tasks of the DU processing, the Forward Error

Correction (FEC) decoding task, and explained the reasons why also general-purpose CPUs can represent a good complement to this HA for the purpose. Moreover, we pioneered the characterization of the utilization of GPUs under real vRAN workloads as a way to justify the sharing and multiplexing opportunities for its computing resources.

In Chapter 4, we have introduced ECORAN, a system that addresses the problems of energy consumption and deployment costs present in new generation vRANs by means of opportunistic offloading of HAs and DU centralization. The solution configures in Near-Real-Time (Near-RT) offloading strategies for tasks of individual DUs sharing a common O-Cloud with HAs and a CPU pool, and relies on a multi-agent contextual bandit algorithm with ideas from mean field theory to improve its convergence and scalability for cost-effective scenarios with several centralized BSs. Using a real O-Cloud platform and traces from a production mobile network, ECORAN provides up to 40% energy savings and up to 60x cost gains with respect to the standard approach used today by the industry.

To increase flexibility and cost-effectiveness of Radio Access Network (RAN) virtualization techniques using Application-Specific Integrated Circuits (ASIC) or Field Programmable Gate Arrays (FPGA) as HA, general-purpose GPU-accelerated computing platforms are being proposed to process the Physical layer (PHY) layer workloads of 5G DUs concurrently with Machine Learning (ML) operations. However, blindly multiplexing available GPU resources using conventional methods can severely disrupt the reliability of inelastic 5G DU workloads, especially when these are balanced with ML elastic tasks. In response to this, in Chapter 5 we have developed YinYangRAN, an innovative O-RAN compliant solution that manages GPU-based HAs, ensuring 5G processing reliability and maximizing the throughput of co-located ML services. Tests with real RAN workloads show that YinYangRAN can potentially reduce the deployment cost by a factor of N compared to the solution using N dedicated GPUs per process, and significantly outperforms traditional GPU multiplexing models by achieving over 50% higher reliability with minimal impact on co-located ML tasks.

In Chapter 6, we have presented CloudRIC, a system that extends the current O-RAN's cloud architecture by implementing an Acceleration Abstraction Layer (AAL) Broker and a Real-Time (RT) RAN Intelligent Controller (RIC) (RT-RIC). These enhancements enable the real-time sharing of HAs across multiple DUs to further reduce deployment costs and the opportunistic usage of CPUs to further reduce the energy consumption without compromising the reliability of the vRAN. The system relies on an implementation based on Data Plane Development Kit (DPDK)'s Baseband Device Library (BBdev) framework and light data-driven models that guarantees low and O-RAN compliant overheads. Thorough empirical evaluations under real-world and worst-case scenarios, we show that CloudRIC delivers substantial advantages with respect to the industry standard approach with 3x and 15x average gains in energy- and cost-efficiency, while maintaining the same 99.999% target reliability.

7.2. Perspectives

The deployment of reliable and inexpensive vRANs with sustainable energetic costs continues to draw significant attention in industry. Therefore, multiple directions for future research are expected to emerge, both in the short term and the long term.

7.2.1. Short-term perspectives

This thesis has demonstrated the significant advantages of sharing and multiplexing computing resources available in heterogeneous cloud platforms that support vRANs. While the focus has primarily been on CPUs and GPUs, there is a clear opportunity for future work to explore other advanced processors that are becoming increasingly prevalent as HAs in industry-grade vRAN deployments, such as FPGAs and ASICs.

In the short term, research can investigate how these processors can be integrated into the existing frameworks to enhance performance and efficiency further. For example, examining the specific workloads that benefit from FPGA reconfigurability or ASIC specialization could yield insights into optimizing resource allocation and reducing latency. Additionally, conducting pilot implementations with industry partners will be essential to validate these methodologies in real-world scenarios, ensuring that the proposed solutions can be effectively deployed and adapted to the dynamic requirements of modern networks.

Moreover, establishing a feedback loop with practitioners can lead to iterative improvements in the proposed approaches, ultimately fostering innovation in vRANs. By actively engaging with stakeholders in the telecommunications industry, this research can contribute to the development of best practices for deploying energy-efficient, cost-effective vRAN solutions that leverage a broader range of computational resources.

7.2.2. Long-term perspectives

The design paradigms proposed in this thesis are not only relevant for current vRAN applications but can also be smoothly adapted to support future traffic demands and emerging HAs. The comprehensive experimental evaluation conducted in Chapter 6, which considers projected vRAN workloads up to 2030 (as detailed in Section 3.1), highlights the forward-thinking nature of this research. The results presented in Figure 6.14 indicate that future HAs are likely to be significantly more performant and energy-efficient than today's CPUs. This insight emphasizes the potential inefficiency of relying solely on traditional CPUs in shared computing platforms of the future.

In the long term, the methodologies and frameworks developed in this thesis can be readily extended to incorporate these future HAs, ensuring that vRAN systems remain agile and capable of meeting evolving demands with target reliability. The data-driven nature of the underlying algorithms allows for seamless integration of new technologies

as they emerge, making it possible to dynamically adapt to advancements in hardware.

Ultimately, this research establishes a robust foundation for developing reliable, cost-effective, and energy-efficient vRANs that are well-equipped to meet the challenges and opportunities of the future of mobile networks. As technologies continue to advance, the frameworks proposed in this thesis can facilitate the exploration of innovative architectures, which are essential for managing the increasing complexity of vRANs.

By maintaining a focus on adaptability and scalability, these frameworks can seamlessly integrate new hardware accelerators as they emerge, ensuring that vRAN systems remain at the forefront of performance and efficiency. Furthermore, fostering ongoing collaboration between academia and industry will be critical in translating these research findings into practical solutions that enhance mobile network capabilities and sustainability. This collaborative approach not only promotes the continuous evolution of vRAN technologies but also drives the development of best practices that can optimize resource utilization and improve overall network performance. In doing so, we can pave the way for a new era of mobile networks that is responsive to the demands of an ever-changing telecommunications landscape.

References

[1] L. L. Schiavo, G. Garcia-Aviles, A. Garcia-Saavedra, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "Cloudric: Open radio access network (o-ran) virtualization with shared heterogeneous computing," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, ser. ACM MobiCom '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 558–572. [Online]. Available: https://doi.org/10.1145/3636534.3649381

[2] L. L. Schiavo, J. A. Ayala-Romero, A. Garcia-Saavedra, M. Fiore, and X. Costa-Perez, "Yinyangran: Resource multiplexing in gpu-accelerated virtualized rans," in *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, 2024, pp. 721–730. [Online]. Available: https://doi.org/10.1109/INFOCOM52122.2024.10621380

[3] J. A. Ayala-Romero, L. Lo Schiavo, A. Garcia-Saavedra, and X. Costa-Perez, "Mean-field multi-agent contextual bandit for energy-efficient resource allocation in vrans," in *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, 2024, pp. 911–920. [Online]. Available: https://doi.org/10.1109/INFOCOM52122.2024.10621197

[4] L. L. Schiavo, G. Garcia-Aviles, A. Garcia-Saavedra, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "Cloudric demo: Open radio access network (o-ran) virtualization with shared heterogeneous computing," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, ser. ACM MobiCom '24. New York, NY, USA: Association for Computing Machinery, 2024, pp. 1–3. [Online]. Available: https://doi.org/10.1145/3636534.3698858

[5] L. Lo Schiavo, M. Fiore, M. Gramaglia, A. Banchs, and X. Costa-Perez, "Forecasting for network management with joint statistical modelling and machine learning," in *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2022, pp. 60–69. [Online]. Available: https://doi.org/10.1109/WoWMoM54355.2022.00028

[6] Deloitte, "The Open Future of Radio Access Networks," *Editorial Report*, 2021.

[7] A. Garcia-Saavedra and X. Costa-PÃ©rez, "O-RAN: Disrupting the Virtualized RAN Ecosystem," *IEEE Communications Standards Magazine*, vol. 5, no. 4, pp. 96–103, 2021.

[8] Heavy Reading, "Heavy Reading's Accelerating Open RAN Platforms Operator Survey," *White Paper*, June 2021.

[9] RCR Wireless News, "From greenfield to brownfield: Open RAN in 2022 (With large scale carrier commitments in place, what's next for the Open RAN ecosystem?)," *Editorial Report*, October 2021.

[10] Analysys Mason, "Open RAN: translating the hype into revenue." Webinar, May 2023.

[11] 3rd Generation Partnership Project (3GPP), "3GPP TR 38.913; Technical Specification Group Radio Access Network; Study on Scenarios and Requirements for Next Generation Access Technologies; (Release 17)," Technical Report, March 2022.

[12] X. Foukas and B. Radunovic, "Concordia: Teaching the 5g vran to share compute," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, ser. SIGCOMM '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 580–596. [Online]. Available: https://doi.org/10.1145/3452296.3472894

[13] G. Garcia-Aviles, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, P. Serrano, and A. Banchs, "Nuberu: Reliable ran virtualization in shared platforms," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 749–761. [Online]. Available: https://doi.org/10.1145/3447993.3483266

[14] Silicom. (2022, Nov.) Silicom's eASIC ACC100 FEC Accelerator. [Online]. Available: https://www.silicom-usa.com/wp-content/uploads/2022/10/Lisbon-ACC100-FEC-Accelerator-Extended-temp-Server-Adapter.pdf

[15] Intel. (2020, September) Enabling 5G Wireless Acceleration in FlexRAN: for the Intel FPGA Programmable Acceleration Card N3000. [Online]. Available: https://www.intel.com/content/www/us/en/programmable/documentation/ocl1575542673666.html

[16] Soma Velayutham. (2019, October) NVIDIA CEO Introduces Aerial – Software to Accelerate 5G on NVIDIA GPUs. [Online]. Available: https://blogs.nvidia.com/blog/2019/10/21/aerial-application-framework-5g-networks/

[17] Keith Dyer. (2023, March) AI everywhere all the time. [Online]. Available: https://the-mobile-network.com/2023/03/ai-everywhere-all-the-time/

[18] A. Kelkar and C. Dick, "Nvidia aerial gpu hosted ai-on-5g," in *2021 IEEE 4th 5G World Forum (5GWF)*, 2021, pp. 64–69. [Online]. Available: https://doi.org/10.1109/5GWF52925.2021.00019

[19] J. Lee, S. Lee, J. Lee, S. D. Sathyanarayana, H. Lim, J. Lee, X. Zhu, S. Ramakrishnan, D. Grunwald, K. Lee, and S. Ha, "Perceive: Deep learning-based cellular uplink prediction using real-time scheduling patterns," in *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 377–390. [Online]. Available: https://doi.org/10.1145/3386901.3388911

[20] A. Banchs, M. Fiore, A. Garcia-Saavedra, and M. Gramaglia, "Network intelligence in 6g: Challenges and opportunities," in *Proceedings of the 16th ACM Workshop on Mobility in the Evolving Internet Architecture*, 2021, pp. 7–12.

[21] J. A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Perez, and G. Iosifidis, "Bayesian online learning for energy-aware resource orchestration in virtualized rans," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[22] J. A. Ayala-Romero, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, A. Banchs, and J. J. Alcaraz, "vrain: A deep learning approach tailoring computing and radio resources in virtualized rans," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–16.

[23] Rethink Technology Research. (2019, November) Nokia says its FPGA strategy hit 5G margins, but other factors are at work too. [Online]. Available: https://rethinkresearch.biz/articles/nokia-says-its-fpga-strategy-hit-5g-margins-but-other-factors-are-at-work-too/

[24] ——. (2020, February) Is general purpose silicon too slow and expensive for the vRAN? [Online]. Available: https://rethinkresearch.biz/articles/is-general-purpose-silicon-too-slow-and-expensive-for-the-vran/

[25] Light Reading. (2022, October) Mavenir unhappy about chip prices for smaller open RAN players. [Online]. Available: https://www.lightreading.com/open-ran/mavenir-unhappy-about-chip-prices-for-smaller-open-ran-players/d/d-id/781327

[26] D. Moolchandani, A. Kumar, and S. R. Sarangi, "Accelerating cnn inference on asics: A survey," *Journal of Systems Architecture*, vol. 113, p. 101887, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762120301612

[27] NVIDIA. (2022, November) NVIDIA V100 Tensor Core GPU Specifications. [Online]. Available: https://images.nvidia.com/content/technologies/volta/pdf/volta-v100-datasheet-update-us-1165301-r5.pdf

[28] C. Jin, X. Bai, C. Yang, W. Mao, and X. Xu, "A review of power consumption models of servers in data centers," *Applied Energy*, vol. 265, p. 114806, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261920303184

[29] Fujitsu, "What is the difference between inline and lookaside accelerators in virtualized distributed units?" *White Paper*, 2023.

[30] J. Ding, R. Doost-Mohammady, A. Kalia, and L. Zhong, "Agora: Real-time massive MIMO baseband processing in software," in *Proceedings of ACM CoNEXT '20*. ACM, 2020.

[31] J. Gong, A. Kalia, and M. Yu, "Scalable Distributed Massive MIMO Baseband Processing," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023, pp. 405–417.

[32] Heavy Reading, "5G Transport: A 2021 Heavy Reading Survey," *White Paper*, February 2022.

[33] N. Docomo, "Base-station equipment with the aim of introducing 3.5-ghz band td-lte," *NTT Docomo Technical Journal*, 2016.

[34] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, "Cloud ran for mobile networks–a technology overview," *IEEE Communications surveys & tutorials*, vol. 17, no. 1, pp. 405–426, 2014.

[35] F. W. Murti, J. A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Pérez, and G. Iosifidis, "An optimal deployment framework for multi-cloud virtualized radio access networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 4, pp. 2251–2265, 2020.

[36] A. Kelkar and C. Dick, "Nvidia aerial gpu hosted ai-on-5g," in *2021 IEEE 4th 5G World Forum (5GWF)*. IEEE, 2021, pp. 64–69.

[37] O-RAN Alliance, "Cloud Architecture and Deployment Scenarios for O-RAN Virtualized RAN (O-RAN.WG6.CADS-v04.00)," Technical Report, October 2022.

[38] E. Dahlman, S. Parkvall, and J. Skold, *5G NR: The next generation wireless access technology*. Academic Press, 2020.

[39] 3GPP, "5G;NR; Physical layer procedures for control," 3GPP TS 38.213 version 16.2.0 Release 16, July 2020.

[40] Y. Blankenship, D. Hui, and M. Andersson, *Channel Coding in NR.* Cham: Springer International Publishing, 2021, pp. 303–332. [Online]. Available: https://doi.org/10.1007/978-3-030-58197-8_10

[41] Intel. (2023) 4th Gen Intel Xeon Scalable Processors with Intel vRAN Boost. Enabling high performance, energy-efficient vRAN with fully integrated acceleration. [Online]. Available: https://download.intel.com/newsroom/2023/5g-communications/2023MWC-vRAN-Fact-Sheet.pdf

[42] AI-RAN Alliance. (2024) Industry Leaders in AI and Wireless Form AI-RAN Alliance. [Online]. Available: https://ai-ran.org/news/industry-leaders-in-ai-and-wireless-form-ai-ran-alliance

[43] T-Mobile. (2024) T–Mobile Announces Technology Partnership with NVIDIA, Ericsson and Nokia to Advance the Future of Mobile Networking with AI at the Center. [Online]. Available: https://www.t-mobile.com/news/business/t-mobile-launches-ai-ran-innovation-center-with-nvidia

[44] O-RAN Alliance, "O-RAN Acceleration Abstraction Layer – General Aspects and Principles (O-RAN.WG6.AAL-GAnP-v04.00)," Technical Specification, October 2022.

[45] S. D'Oro, L. Bonati, M. Polese, and T. Melodia, "Orchestran: Network automation through orchestrated intelligence in the open ran," in *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022, pp. 270–279. [Online]. Available: https://doi.org/10.1109/INFOCOM48880.2022.9796744

[46] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Colo-ran: Developing machine learning-based xapps for open ran closed-loop control on programmable experimental platforms," *IEEE Transactions on Mobile Computing*, pp. 1–14, 2022. [Online]. Available: https://doi.org/10.1109/TMC.2022.3188013

[47] J. A. Ayala-Romero *et al.*, *EdgeBOL: Automating Energy-Savings for Mobile Edge AI.* New York, NY, USA: Association for Computing Machinery, 2021, pp. 397–410. [Online]. Available: https://doi.org/10.1145/3485983.3494849

[48] Intel, "Virtual RAN with Hardware Acceleration," *White Paper*, 2020.

[49] Dell, "Dell Open RAN Accelerator Card," *Solution brief*, 2022.

[50] J. Xing, J. Gong, X. Foukas, A. Kalia, D. Kim, and M. Kotaru, "Enabling resilience in virtualized rans with atlas," in *Proceedings of the 29th Annual International*

Conference on Mobile Computing and Networking, ser. ACM MobiCom '23. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3570361.3613276

[51] W.-H. Ko, U. Ghosh, U. Dinesha, R. Wu, S. Shakkottai, and D. Bharadia, "EdgeRIC: Empowering real-time intelligent optimization and control in NextG cellular networks," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. Santa Clara, CA: USENIX Association, Apr. 2024, pp. 1315–1330. [Online]. Available: https://www.usenix.org/conference/nsdi24/presentation/ko

[52] O. Adamuz-Hinojosa, L. Zanzi, V. Sciancalepore, A. Garcia-Saavedra, and X. Costa-Perez, "Oranus: Latency-tailored orchestration via stochastic network calculus in 6g o-ran," in *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, 2024, pp. 61–70. [Online]. Available: https://doi.org/10.1109/INFOCOM52122.2024.10621356

[53] F. Mungari, C. Puligheddu, A. Garcia-Saavedra, and C. F. Chiasserini, "Oreo: O-ran intelligence orchestration of xapp-based network services," in *IEEE INFOCOM 2024 - IEEE Conference on Computer Communications*, 2024, pp. 71–80. [Online]. Available: https://doi.org/10.1109/INFOCOM52122.2024.10621166

[54] G. Auer *et al.*, "How much energy is needed to run a wireless network?" *IEEE wireless communications*, vol. 18, no. 5, pp. 40–49, 2011.

[55] J. Lorincz, T. Garma, and G. Petrovic, "Measurements and modelling of base station power consumption under real traffic loads," *Sensors*, vol. 12, no. 4, pp. 4281–4310, 2012.

[56] O. Arnold *et al.*, "Power consumption modeling of different base station types in heterogeneous cellular networks," in *2010 Future Network & Mobile Summit*. IEEE, 2010, pp. 1–8.

[57] J. G. Andrews *et al.*, "What will 5g be?" *IEEE Journal on selected areas in communications*, vol. 32, no. 6, pp. 1065–1082, 2014.

[58] J. A. Ayala-Romero *et al.*, "Experimental evaluation of power consumption in virtualized base stations," in *IEEE ICC '21*, 2021.

[59] J. A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Perez, and G. Iosifidis, "Orchestrating Energy-Efficient vRANs: Bayesian Learning and Experimental Results," *IEEE Transactions on Mobile Computing*, vol. 22, no. 5, pp. 2910–2924, 2023.

[60] R. Lowe *et al.*, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[61] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *International conference on machine learning*. PMLR, 2018, pp. 5571–5580.

[62] M. Li, Z. Qin, Y. Jiao, Y. Yang, J. Wang, C. Wang, G. Wu, and J. Ye, "Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning," in *The world wide web conference*, 2019, pp. 983–994.

[63] D. Chen *et al.*, "Mean field deep reinforcement learning for fair and efficient uav control," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 813–828, 2020.

[64] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 1774–1783.

[65] N. Zhao *et al.*, "Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 11, pp. 5141–5152, 2019.

[66] F. B. Mismar, J. Choi, and B. L. Evans, "A framework for automated cellular network tuning with reinforcement learning," *IEEE Transactions on Communications*, vol. 67, no. 10, pp. 7152–7167, 2019.

[67] Z. Zhang *et al.*, "DQ scheduler: Deep reinforcement learning based controller synchronization in distributed SDN," in *IEEE ICC 2019*, 2019, pp. 1–7.

[68] X. Wang, X. Guo, J. Chuai, Z. Chen, and X. Liu, "Kernel-based multi-task contextual bandits in cellular network configuration," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 1517–1526.

[69] J. Chuai, Z. Chen, G. Liu, X. Guo, X. Wang, X. Liu, C. Zhu, and F. Shen, "A collaborative learning based approach for parameter configuration of cellular networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1396–1404.

[70] J. A. Ayala-Romero *et al.*, "Online learning for energy saving and interference coordination in hetnets," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1374–1388, 2019.

[71] J. Li, D. Ferry, S. Ahuja, K. Agrawal, C. Gill, and C. Lu, "Mixed-criticality federated scheduling for parallel real-time tasks," *Real-time systems*, vol. 53, no. 5, pp. 760–811, 2017.

[72] G. Prekas, M. Kogias, and E. Bugnion, "Zygos: Achieving low tail latency for microsecond-scale networked tasks," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 325–341.

[73] H. Qin, Q. Li, J. Speiser, P. Kraft, and J. Ousterhout, "Arachne:{Core-Aware} thread management," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018, pp. 145–160.

[74] K. Kaffes, T. Chong, J. T. Humphries, A. Belay, D. Mazières, and C. Kozyrakis, "Shinjuku: Preemptive scheduling for {μsecond-scale} tail latency," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, 2019, pp. 345–360.

[75] A. Ousterhout, J. Fried, J. Behrens, A. Belay, and H. Balakrishnan, "Shenango: Achieving high {CPU} efficiency for latency-sensitive datacenter workloads," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, 2019, pp. 361–378.

[76] M. Marty, M. de Kruijf, J. Adriaens, C. Alfeld, S. Bauer, C. Contavalli, M. Dalton, N. Dukkipati, W. C. Evans, S. Gribble *et al.*, "Snap: A microkernel approach to host networking," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, 2019, pp. 399–413.

[77] B. Li, V. Gadepally, S. Samsi, and D. Tiwari, "Characterizing multi-instance gpu for machine learning workloads," in *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2022, pp. 724–731.

[78] B. Li, T. Patel, S. Samsi, V. Gadepally, and D. Tiwari, "Miso: exploiting multi-instance gpu capability on multi-tenant gpu clusters," in *Proceedings of the 13th Symposium on Cloud Computing*, 2022, pp. 173–189.

[79] P. Yu and M. Chowdhury, "Fine-grained gpu sharing primitives for deep learning applications," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 98–111, 2020.

[80] B. Wu, Z. Zhang, Z. Bai, X. Liu, and X. Jin, "Transparent {GPU} sharing in container clouds for deep learning workloads," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023, pp. 69–85.

[81] R. Falkenberg and C. Wietfeld, "FALCON: An accurate real-time monitor for client-based mobile network data analytics," in *2019 IEEE Global Communications Conference (GLOBECOM)*. Waikoloa, Hawaii, USA: IEEE, Dec. 2019. [Online]. Available: https://arxiv.org/abs/1907.10110

[82] N. Ludant, P. Robyns, and G. Noubir, "From 5g sniffing to harvesting leakages of privacy-preserving messengers," in *2023 2023 IEEE Symposium on Security and Privacy (SP) (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2023, pp. 3146–3161. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP46215.2023.00110

[83] P. F. Pérez, C. Fiandrino, and J. Widmer, "Characterizing and modeling mobile networks user traffic at millisecond level," in *Proceedings of the 17th ACM Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, ser. WiNTECH '23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 64–71. [Online]. Available: https://doi.org/10.1145/3615453.3616509

[84] Ericsson, "Mobility Report," *White Paper*, June 2022.

[85] Light Reading. (2023, August) Chip choices kickstart open RAN war between lookaside and inline. [Online]. Available: https://www.lightreading.com/semiconductors/chip-choices-kickstart-open-ran-war-between-lookaside-and-inline

[86] G. Falcao, L. Sousa, and V. Silva, "Massively ldpc decoding on multicore architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 2, pp. 309–322, 2010.

[87] Intel. (2019, May) FlexRAN LTE and 5G NR FEC Software Development Kit Modules. [Online]. Available: https://software.intel.com/content/www/us/en/develop/articles/flexran-lte-and-5g-nr-fec-software-development-kit-modules.html

[88] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. 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*, 2016, pp. 25–32.

[89] W. T. Han and R. Knopp, "OpenAirInterface: A pipeline structure for 5G," in *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*. IEEE, 2018, pp. 1–4.

[90] DPDK. (2023) Wireless Baseband Device Library. [Online]. Available: https://doc.dpdk.org/guides/prog_guide/bbdev.html

[91] O-RAN Alliance. (2022) O-DU Low Project Introduction. [Online]. Available: https://docs.o-ran-sc.org/projects/o-ran-sc-o-du-phy/en/latest/overview1.html

[92] 3rd Generation Partnership Project (3GPP), "3GPP TS 38.214; Technical Specification Group Radio Access Network; NR; Physical layer procedures for data (Release 16)," Technical Specification, October 2021.

[93] A. Udalcovs, M. Levantesi, P. Urban, D. A. Mello, R. Gaudino, O. Ozolins, and P. Monti, "Total cost of ownership of digital vs. analog radio-over-fiber architectures for 5g fronthauling," *IEEE Access*, vol. 8, pp. 223 562–223 573, 2020.

[94] O-RAN Alliance, "O-RAN Working Group 4 (Open Fronthaul Interfaces WG). Control, User and Synchronization Plane Specification (O-RAN.WG4.CUS.0-v10.00)," Technical Report, October 2022.

[95] G. O. Pérez, D. L. Lopez, and J. A. Hernández, "5g new radio fronthaul network design for ecpri-ieee 802.1 cm and extreme latency percentiles," *IEEE Access*, vol. 7, pp. 82 218–82 230, 2019.

[96] M. T. Arefin and F. R. Reza, *AWGN & Rayleigh Fading Channel: A Throughput Analysis for Reliable Data Transmission.* VDM Verlag Dr. Muller, 2011.

[97] M. Feng, S. Mao, and T. Jiang, "Base station on-off switching in 5g wireless networks: Approaches and challenges," *IEEE Wireless Communications*, vol. 24, no. 4, pp. 46–54, 2017. [Online]. Available: https://doi.org/10.1109/MWC.2017.1600353

[98] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *Advances in neural information processing systems*, vol. 29, 2016.

[99] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *PMLR 2014*, 2014, pp. 387–395.

[100] C. Domb, *Phase transitions and critical phenomena.* Elsevier, 2000.

[101] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[102] J. A. Ayala-Romero, P. Mernyei, B. Shi, and D. Mazón, "Kraf: A flexible advertising framework using knowledge graph-enriched multi-agent reinforcement learning," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 47–56.

[103] Z. Guan, H. Wu, Q. Cao, H. Liu, W. Zhao, S. Li, C. Xu, G. Qiu, J. Xu, and B. Zheng, "Multi-agent cooperative bidding games for multi-objective optimization in e-commercial sponsored search," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 2899–2909.

[104] J. Li, K. Kuang, B. Wang, F. Liu, L. Chen, F. Wu, and J. Xiao, "Shapley counterfactual credits for multi-agent reinforcement learning," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 934–942.

[105] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *International conference on machine learning*. PMLR, 2019, pp. 2961–2970.

[106] D. Bertsekas, *A Course in Reinforcement Learning*. Athena Scientific, 2023.

[107] ——, *Dynamic programming and optimal control: Volume I*. Athena scientific, 2012, vol. 4.

[108] P. J. Huber, "Robust estimation of a location parameter," *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518, 1992.

[109] Y.-s. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "Ecf: An mptcp path scheduler to manage heterogeneous paths," in *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 147–159. [Online]. Available: https://doi.org/10.1145/3143361.3143376

[110] Luca Cominardi. (2021, July) Zenoh performance: a stroll in Rust async wonderland. [Online]. Available: https://zenoh.io/blog/2021-07-13-zenoh-performance-async/

[111] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 1861–1870. [Online]. Available: https://proceedings.mlr.press/v80/haarnoja18b.html

[112] P. Vicente, P. M. Santos, B. Asulba, N. Martins, J. Sousa, and L. Almeida, "Comparing performance of machine learning tools across computing platforms," in *2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS)*. IEEE, 2023, pp. 1185–1189.