

SUSTAINABLE OPERATION OF 5G VIRTUALIZED RAN  
COMPUTING INFRASTRUCTURE

by

NIKOLAOS APOSTOLAKIS

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. Albert Banchs

Dr. Marco Gramaglia

Tutor:

Dr. Albert Banchs

September 2025



---

*Sustainable Operation of 5G Virtualized RAN Computing Infrastructure*

Prepared by:

Nikolaos Apostolakis

Telematic Engineering Department, Universidad Carlos III de Madrid

IMDEA Networks Institute

Contact: n [dot] apostolakis [at] outlook [dot] com

Under the advice of:

Dr. Albert Banchs

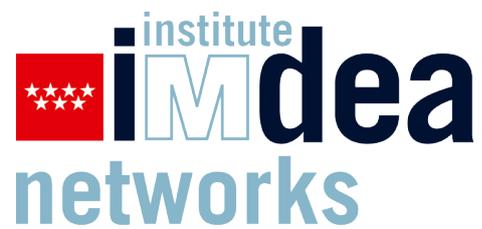
Telematic Engineering Department, Universidad Carlos III de Madrid

IMDEA Networks Institute

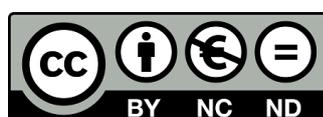
Dr. Marco Gramaglia

Telematic Engineering Department, Universidad Carlos III de Madrid

This work has been supported by:



The content of this thesis is distributed under license  
"Creative Commons Attribution - Non Commercial - Non Derivatives"



*In every error and every challenge, there  
lies not defeat, but the path to progress —  
πάθει μάθος.*

*– Αισχύλος, Άγαμέμνων*



# Acknowledgements

---

First and foremost, I would like to express my deepest and most heartfelt gratitude to my advisors, Professor Albert Banchs and Professor Marco Gramaglia, for their continuous support throughout these years. Thank you for allowing me to pursue my passion and for guiding me through the early steps of my research career. Your trust, encouragement, and expertise have been instrumental in shaping my scientific mindset and helping me grow into an independent researcher. I am especially grateful for the freedom you gave me to explore diverse, cutting-edge topics— an experience that has greatly enriched my technical and theoretical skills and prepared me for the challenges ahead.

It has also been a great pleasure to work closely with the distinguished researchers Dr. Andres Garcia-Saavedra, Dr. Livia Elena Chatzieftheriou, Dr. Dario Bega, Marta Sierra-Obea, Dr. A. Ayala-Romero, and Professor Marco Fiore. I am deeply grateful for our fruitful collaborations—your insightful feedback has helped shape my research perspective, and our teamwork has resulted in significant scientific contributions. I would also like to extend my sincere thanks to Professor Pablo Serrano, Dr. Tejas Subramanya, Dr. Henning Sanneck, and Dr. Xavier Costa-Perez for the impactful collaborations we shared, which led to important findings and advancements.

I would like to sincerely thank Dr. Christian Ibars Casas for hosting me at NVIDIA, California in 2024. Working at an organization that is revolutionizing the computing paradigm across various domains, including telecommunications, was an eye-opening and formative experience that would not have happened without Christian’s belief in me. I am also deeply grateful to my mentor, Harsha Banuli, who supported me throughout my stay and taught me how to design systems at scale, as well as to Curtis Rhodes for his continuous guidance and valuable input in software design.

In this direction, I would also like to thank IMDEA Networks Institute as a whole. From the IT department to the faculty members, you all contributed to making this experience truly memorable. Thank you for creating an environment that fosters collaboration and excellence, and for providing the support and tools necessary to conduct world-class research.

Throughout my PhD years, life in Madrid has gifted me with countless unforgettable

experiences and lifelong friendships. First and foremost, I could never forget my best friend and now PhD holder, Manolis (*el carpintero*), with whom I shared countless moments, from spontaneous guitar sessions on random Wednesday nights to deep conversations about our doctoral journeys. These talks helped me ease my worries and will continue to shape me for the rest of my life. I also want to thank Panos, the very first person I met when I moved to Spain, for being there every step of the way, unconditionally and supportively. In this spirit, I also owe a big *grazie* to Mauro, my PhD and office deskmate, who has listened to my deepest confessions and has witnessed my highest highs (and occasionally lows). My stay in California would not have been the same without my colleague Giannis and flatmates George and Kona – thank you for spending this summer with me and building this strong friendship.

A big thank you goes to the Greek community in Madrid (Thanos, Mike, Dionysis, Giannis, Panagiotis, Theo) for the endless laughs that made this city feel like home. A special shoutout to Aris and Maimu, the bar that somehow adopted me since my first month in Spain, I am still not sure how that happened, but I am glad it did. And of course, our beloved music group, *Madrikeniki Compania*, which was not just a band playing in Maimu, but free therapy disguised as *rempetiko*, which turned out to be the perfect antidote to the stressful academic deadlines. Of course, I can not forget all my very good friends from Athens (Thanassis, Giorgos, Vasilis, Giannis, Kosmas, Stella, Loukas) and my undergrad days (Christos, Kostas, Panos, Stavros, Thymios) – all of them were truly supportive in this endeavor.

I cannot help but dedicate a very special thank you to my partner, Alexandra (or Alie, or Alex, depending on the day). You've been there from the very start of this journey, always by my side with patience, strength, and unwavering love. Thank you for being my stand during the most stressful moments, for believing in me even when I doubted myself, for not letting me give up, and for all the personal sacrifices you've made so I could chase this dream. This achievement is as much yours as it is mine, and I am endlessly grateful to share it with you.

Wrapping up my acknowledgements, I want to say a special and most sincere ευχαριστώ (efcharistó) to my parents and siblings. Ξεκινάω από τον αδερφό μου, Μηνά, που με στήριξε σε όλο αυτό το ταξίδι από την 1η μέρα όταν έφυγα το 2020, έχοντας και ο ίδιος φύγει στο εξωτερικό. Στην αδερφή μου, Ευγενία, που κάθε φορά ανυπομονεί πότε θα γυρίσω στην Ελλάδα και ήταν η πρώτη που με ενθάρρυνε να ξεκινήσω. Αλλά κυρίως, στους γονείς μου, Κώστα και Ελισάβετ, που μου έδωσαν ανιδιοτελή αγάπη και στήριξη, με μεγάλωσαν και γαλούχησαν με τις πιο σωστές αξίες, μου μεταλαμπάδευσαν την σημασία της σκληρής δουλειάς και την περιέργεια της εξερεύνησης, που με προέτρεπαν πάντα να βγάζω τον καλύτερο μου εαυτό και που ήταν πάντα εκεί, ένα τηλέφωνο μακριά, ανεξαρτήτων συνθηκών.

# Published Content

---

This thesis is based on the following published papers:

[1] **Nikolaos Apostolakis**, Marco Gramaglia and Pablo Serrano, "Design and Validation of an Open Source Cloud Native Mobile Network," in *IEEE Communications Magazine*, vol. 60, no. 11, pp. 66-72, November 2022, <https://doi.org/10.1109/MCOM.003.2200195>.

- This work is fully included and the contents are reported in Chapter 2 and 3.
- The thesis author led the conceptual design of the cloud-native 5G core design, led the implementation of the solution, ran the experiments, generated plots, and wrote several parts of the manuscript.
- The material from this source included in this thesis is not singled out with typographic means and references.
- Q1 in Telecommunications with an impact factor of 8.2, according to the Clarivate Journal Citation Reports.

[2] **Nikolaos Apostolakis**, Marco Gramaglia, Livia Elena Chatzieleftheriou, Tejas Subramanya, Albert Banchs and Henning Sanneck, "ATHENA: Machine Learning and Reasoning for Radio Resources Scheduling in vRAN Systems," in *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 2, pp. 263-279, Feb. 2024, <https://doi.org/10.1109/JSAC.2023.3336155>.

- This work is fully included and the contents are reported in Chapters 2, 4 and 5.
- The thesis author is the first author of this work and led the design and architecture of ATHENA framework, its implementation and experimentation in hardware, and participated in writing several parts of the manuscript.
- The material from this source included in this thesis is not singled out with typographic means and references.
- Q1 in Telecommunications with an impact factor of 17.2, according to the Clarivate Journal Citation Reports.

[3] **Nikolaos Apostolakis**, Livia Elena Chatzieleftheriou, Dario Bega, Marco Gramaglia and Albert Banchs, "Digital Twins for Next-Generation Mobile Networks: Applications

and Solutions," in IEEE Communications Magazine, vol. 61, no. 11, pp. 80-86, November 2023, doi: <https://doi.org/10.1109/MCOM.001.2200854>.

- This work is fully included and the contents are reported in Chapters 2 and 5.

- The thesis author led the conceptual design of the digital twin, led the implementation of the solution, ran the experiments, generated the plots, and wrote several parts of the manuscript.

- The material from this source included in this thesis is not singled out with typographic means and references.

- Q1 in Telecommunications with an impact factor of 8.2, according to the Clarivate Journal Citation Reports.

[4] **Nikolaos Apostolakis**, Marta Sierra-Obea, Marco Gramaglia, Jose A. Ayala-Romero, Andres Garcia-Saavedra, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. 2025. Quantum Computing in the RAN with Qu4Fec: Closing Gaps Towards Quantum-based FEC Processors. *Proc. ACM Meas. Anal. Comput. Syst.* 9, 2, Article 36 (June 2025), 25 pages. <https://doi.org/10.1145/3727128>.

- This work is fully included and the contents are reported in Chapters 2 and 6.

- The thesis author led the conceptual design of Qu4Fec, led the derivation of the first QUBO formulation, ran the experiments on Simulated and Quantum Annealers, identified the code design limitations of QBP and hardware limitations of Quantum Annealers, took part in generating plots, and writing several parts of the manuscript.

- The material from this source included in this thesis is not singled out with typographic means and references.

- CSrankings-listed conference and rated as an A\* conference according to the CORE 2023 classification.

[5] **Nikolaos Apostolakis**, Marta Sierra-Obea, Marco Gramaglia, Jose A. Ayala-Romero, Andres Garcia-Saavedra, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. 2025. Quantum Computing in the RAN with Qu4Fec: Closing Gaps Towards Quantum-based FEC Processors. In *Abstracts of the 2025 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS/PERFORMANCE '25)*. Association for Computing Machinery, New York, NY, USA, 59-60. <https://doi.org/10.1145/3652963.3655054>.

- This work is fully included and the contents are reported in Chapters 2 and 6.

- This work is the conference abstract version of publication [4].

- CSrankings-listed conference and rated as an A\* conference according to the CORE 2023 classification.

# Abstract

---

Virtualized Radio Access Networks (vRANs) are undergoing a significant paradigm shift, moving from specialized, vendor-specific hardware platforms to commercial off-the-shelf (COTS) servers for baseband processing and upper-layer 5G functions. This transition enables operators to avoid vendor lock-in and promotes interoperability across diverse software and hardware ecosystems. Moreover, shared COTS infrastructure allows for efficient multi-tenant resource usage, simplified orchestration, and ultimately, reduced capital (CapEx) and operational (OpEx) expenditures.

However, typical vRAN implementations, particularly the physical (PHY) layer running on COTS servers, struggle to meet the computational demands and strict timing requirements of 5G. While modern CPUs support both data and pipeline-level parallelism, they are often unable to handle the millisecond-scale latencies required for efficient 5G operation. To overcome these limitations, vRAN deployments increasingly rely on specialized hardware accelerators such as Graphics Processing Units (GPUs), Application-Specific Integrated Circuits (ASICs), and Field Programmable Gate Arrays (FPGAs), which are better equipped to meet these real-time constraints and deliver the necessary quality of service.

Yet, these solutions introduce their challenges in terms of sustainability, flexibility, and manageability. GPUs, while easier to orchestrate in cloud-native environments, require substantial upfront investment and consume significantly more power than COTS servers. Although vendors aim to amortize GPU costs by colocating high-revenue stream workloads like Artificial Intelligence (AI) inference and training, this strategy remains largely theoretical and unproven in practice. ASICs and FPGAs, on the other hand, are often the default choice for many vRAN vendors due to their lower energy consumption and cost-effectiveness. However, their lack of reprogrammability and limited support for multi-tenant sharing hinder efficient management and orchestration by end-to-end network controllers.

The aforementioned limitations of external hardware in supporting the 5G vRAN vision highlight the need for financially sustainable solutions based on CPU-only COTS servers, resources that are inherently easier to share, manage, and maintain across the evolution of mobile networks. At the same time, there is a growing demand for new

computing paradigms that can improve hardware resource utilization and reduce the energy footprint of telecom operators. This thesis contributes to both of these directions.

First, it introduces the design and implementation of a CPU-aware Medium Access Control (MAC) scheduler that intelligently allocates computational resources to users, significantly enhancing the reliability of time-critical task execution. Second, it reformulates the Forward Error Correction (FEC) problem, an essential component of 5G and 6G baseband processing, as an optimization problem that can be efficiently solved on a quantum computer. This positions quantum hardware as a promising addition to the data center resource pool, offering a potentially lower energy footprint compared to conventional accelerators.

The first part of this thesis introduces **ATHENA**, a Machine Learning (ML)-based radio resource controller that intelligently allocates uplink grants to users based on the CPU quota available to the virtualized Distributed Unit (vDU). The vDU typically operates alongside third-party applications on conventional shared COTS infrastructure, eliminating the need for external accelerator units. **ATHENA** is deployed as an O-RAN real-time application (dApp) within the O-RAN Real-Time RIC (Radio Intelligent Controller) and leverages instantaneous CPU congestion metrics provided by the Service Management and Orchestration (SMO) framework. These metrics capture the effects of resource contention (commonly referred to as the noisy neighbor effect) caused by non-RAN applications.

By dynamically controlling radio resource allocation in response to CPU load, **ATHENA** improves the determinism of vDU processing timelines, enabling them to meet the strict latency requirements imposed by 5G technologies and use cases. **ATHENA** is built on a Reinforcement Learning (RL) framework and is platform-agnostic as it does not assume any specific CPU architecture or configuration. Compared to baseline schedulers, it delivers higher reliability under both low and high levels of CPU contention from neighboring applications.

The development and evaluation of **ATHENA** were based on the design and implementation of a fully cloud-native, end-to-end 5G deployment, encompassing both RAN and core network components. This system is rigorously described in the current thesis, and to the best of the author's knowledge, at the time of its implementation, it was the first open-source initiative to provide such a complete system without requiring expensive hardware RF front-ends. This approach democratizes access to state-of-the-art 5G deployments, supporting researchers and practitioners in developing algorithms and identifying challenges associated with CPU-only vDU architectures.

In a direction orthogonal to conventional processor-based approaches, the second part of this thesis introduces **Qu4Fec**, a novel LDPC (Low-Density Parity-Check) decoder implemented on quantum computers. **Qu4Fec** reformulates LDPC decoding as an optimization problem that can be natively solved on quantum hardware. In

simulation, **Qu4Fec** outperforms existing state-of-the-art approaches by 1.82 dB and underscores the importance of carefully designing LDPC codes to preserve their error-correcting performance within the quantum paradigm. Moreover, **Qu4Fec** achieves comparable decoding performance to conventional LDPC decoding algorithms, suggesting the feasibility of offloading 5G FEC tasks to quantum computers, which, according to recent studies, may offer a lower energy footprint than conventional computing platforms, contributing in this way to the sustainability of mobile systems.

However, experimental evaluations on actual quantum hardware revealed significant discrepancies compared to simulation results. These were primarily attributed to the high noise levels in current quantum systems, as well as limitations introduced by scaling and quantization during quantum program compilation. Finally, this thesis offers a forward-looking perspective on the use of quantum machines for mobile network workloads and proposes hardware-level enhancements to future quantum architectures that could better support LDPC decoding tasks.



# Table of Contents

---

<b>Acknowledgements</b>	<b>vii</b>
<b>Published Content</b>	<b>ix</b>
<b>Abstract</b>	<b>xi</b>
<b>Table of Contents</b>	<b>xv</b>
<b>List of Tables</b>	<b>xix</b>
<b>List of Figures</b>	<b>xxi</b>
<b>List of Acronyms</b>	<b>xxiii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	2
1.2 Research Challenges . . . . .	4
1.3 Contributions . . . . .	6
1.4 Outline . . . . .	9
<b>2 Background</b>	<b>11</b>
2.1 5G NR . . . . .	11
2.1.1 LDPC Codes . . . . .	14
2.1.2 MAC Scheduler . . . . .	16
2.1.3 Digital Twin in 5G . . . . .	17
2.1.4 AI in 5G . . . . .	18
2.2 Quantum Computing . . . . .	19
2.2.1 Quantum Annealers . . . . .	19
2.2.2 Quantum RAN Power Analysis . . . . .	22
2.3 Related Work . . . . .	22
2.3.1 Mobile Networking Software . . . . .	22
2.3.2 AI/ML MAC Scheduling . . . . .	23

2.3.3	CPU-Aware RAN . . . . .	23
2.3.4	Quantum Baseband Processing . . . . .	25
<b>3</b>	<b>Orchestration of Cloud-Native Mobile Networks</b>	<b>27</b>
3.1	Cloud-Native Core and RAN Design . . . . .	27
3.1.1	Management and Orchestration . . . . .	28
3.1.2	Core Network . . . . .	28
3.1.3	RAN . . . . .	29
3.1.4	Channel Emulation . . . . .	30
3.1.5	Monitoring . . . . .	30
3.2	Network Deployment . . . . .	32
3.2.1	Kubernetes Manifests . . . . .	32
3.2.2	Helm Charts . . . . .	33
3.3	Experiments . . . . .	34
3.3.1	Dynamic NF orchestration . . . . .	35
3.3.2	UE Mobility . . . . .	36
<b>4</b>	<b>ATHENA: Resource-Aware MAC Scheduling</b>	<b>39</b>
4.1	System model . . . . .	40
4.1.1	Model Components . . . . .	40
4.1.2	Problem: Resource Scheduling in Congested vRANs . . . . .	41
4.1.3	Relevance for Uplink-Intensive Use Cases . . . . .	43
4.2	ATHENA . . . . .	44
4.2.1	ATHENA-ML Components . . . . .	45
4.2.2	ATHENA-ML Internal Design . . . . .	46
4.3	Implementation . . . . .	49
4.3.1	ATHENA-ML Integration into vRAN Software . . . . .	50
4.3.2	ATHENA-ML Pipeline . . . . .	50
4.3.3	Multi-User Scenario . . . . .	53
4.4	Evaluation . . . . .	54
4.4.1	ATHENA-ML Convergence . . . . .	54
4.4.2	ATHENA-ML Performance . . . . .	55
4.4.3	Advantages of Joint MCS-PRB Management . . . . .	57
4.4.4	Multi-User Scenario . . . . .	58
<b>5</b>	<b>Explainability and Twinning for AI on RAN</b>	<b>61</b>
5.1	Digital Twin . . . . .	62
5.1.1	Design . . . . .	62
5.1.2	Model . . . . .	63
5.1.3	Evaluation . . . . .	64

5.2	Explainable AI . . . . .	66
5.2.1	Attributive Explanations . . . . .	66
5.2.2	Contrastive Explanations . . . . .	67
5.2.3	Actionable Explanations . . . . .	68
<b>6</b>	<b>Qu4Fec: LDPC Decoding on Quantum Platforms</b>	<b>71</b>
6.1	Decoding in the Quantum Environment . . . . .	73
6.1.1	LDPC QUBO Formulation . . . . .	73
6.1.2	Code Design . . . . .	74
6.1.3	Qu4Fec: Alternative QUBO Formulation . . . . .	76
6.1.4	Stability in Qu4Fec . . . . .	79
6.2	Simulated and Experimental Evaluations . . . . .	80
6.2.1	Benchmarking Qu4Fec via Simulated Annealing . . . . .	81
6.2.2	Effects of Non-ideal Embeddings . . . . .	82
6.2.3	Scaling and Quantization . . . . .	84
6.2.4	Effects on the QA process . . . . .	86
6.3	Towards Quantum-based Baseband Processors . . . . .	87
6.3.1	Embedding Algorithms Analysis . . . . .	87
6.3.2	Optimal QPU Graph Structure . . . . .	87
6.3.3	Zephyr Layout Parameterization . . . . .	88
6.3.4	QPU Fabric Extensions . . . . .	89
<b>7</b>	<b>Conclusions and Perspectives</b>	<b>91</b>
7.1	Conclusions . . . . .	91
7.2	Future Perspectives . . . . .	93
	<b>References</b>	<b>95</b>



# List of Tables

---

1.1	Software released as open source. . . . .	8
1.2	Mapping of research challenges to contributions, chapters and papers. . .	9
2.1	Mapping of related work and its limitations to the thesis contributions. .	26
4.1	ATHENA notation table. . . . .	40



# List of Figures

---

1.1	vDU hardware resources. . . . .	2
1.2	Thesis contributions. . . . .	6
2.1	Uplink Processing Pipeline in 5G NR. . . . .	13
2.2	Interaction of Physical and Digital Twins in a service provider context. . .	17
2.3	Machine Reasoning in network management and orchestration. . . . .	18
2.4	Embedding Visualization. . . . .	21
3.1	Cloud-Native end-to-end architecture . . . . .	28
3.2	Core and RAN Infrastructure setup. . . . .	29
3.3	The evaluated mobility and load scenarios between UEs and eNBs. The UEs connect with the eNBs through the GRC Broker, that enables intra- or inter-eNB handovers. The eNBs/gNBs connect to the Open5Gs. . . . .	31
3.4	The network functions included in Open5Gs and srsRAN. . . . .	32
3.5	CPU Utilization of two 5G core instances. . . . .	36
3.6	RAN throughput. . . . .	36
4.1	Decoding time under different SNR, MCS, PRB combinations, and congestion factors. . . . .	42
4.2	ATHENA and its integration with 3GPP and O-RAN architectures. . . .	43
4.3	Actor's objective across training epochs. . . . .	55
4.4	Throughput served by a gNB running ATHENA-ML, ATHENA-MCS and the Baseline scheduling algorithm. . . . .	56
4.5	Reliability of ATHENA-ML, ATHENA-MCS and Baseline scheduling algorithms, for different congestions factors. . . . .	56
4.6	Decoding time and MCS distribution for ATHENA-ML and Baseline scheduling algorithms under different congestion factors. . . . .	56
4.7	MCS and PRB decisions of ATHENA-ML and ATHENA-MCS. . . . .	57
4.8	Throughput for different congestion factors, user load and cell utilization. . . . .	58
5.1	Building blocks of digital twin. . . . .	63

5.2	PDF of predicted decoding times and decoding probability prediction. . .	64
5.3	DT validation loss under distribution changes. . . . .	65
5.4	ATHENA-ML throughput, distilled decision tree and critic's reward prediction. . . . .	67
5.5	Congestion factor re-orchestration by ATHENA-MR to satisfy KPI. . . .	69
6.1	Parity Check Matrix and Tanner Graph. . . . .	75
6.2	Steps from designing an LDPC code toward Quantum-based decoding. . .	76
6.3	BER, BLER and cycle length distribution comparison of QBP and Gallager code design methods. . . . .	77
6.4	Sample comparison of QBP and Qu4Fec energy levels. . . . .	79
6.5	BER/BLER curves of Qu4Fec and QBP using different code design for a (2,3,420) LDPC code. . . . .	82
6.6	BER/BLER curves for BP, Qu4Fec and QBP and different code lengths. .	83
6.7	Qubits' chain length distribution using different embedding methods. . . .	84
6.8	Original, scaled linear, and quadratic term histogram for the Qu4Fec and Qu4Fec-unweighted QUBO formulations. . . . .	85
6.9	Percentage change of linear and quadratic terms due to scaling and quantization effect for Qu4Fec and Qu4Fec-unweighted formulations under 3 different bit widths. . . . .	85
6.10	Percentage energy discrepancy between QA and SA for Qu4Fec and Qu4Fec-unweighted QUBO formulations. . . . .	87
6.11	Maximum chain length achieved by MM for different code word lengths and target QPU architecture. . . . .	88
6.12	Effect of $m$ and $t$ on the maximum chain length . . . . .	89
6.13	Higher order couplers on Zephyr qubit lattice and their effect on maximum chain length. . . . .	90

# List of Acronyms

---

<b>RAN</b>	Radio Access Network
<b>Near-RT RIC</b>	Near Real-time RAN Intelligent Controller
<b>Non-RT RIC</b>	Non Real-time RAN Intelligent Controller
<b>O-Cloud</b>	O-RAN Cloud
<b>SMO</b>	Service Management and Orchestration Framework
<b>BER</b>	Bit Error Rate
<b>BLER</b>	Block Error Rate
<b>BCE</b>	Binary Cross Entropy
<b>NLL</b>	Negative Log-Likelihood
<b>CDF</b>	Cumulative Distribution Function
<b>PDF</b>	Probability Density Function
<b>KPI</b>	Key Performance Indicator
<b>vRAN</b>	Virtualized Radio Access Network
<b>vDU</b>	Virtualized DU
<b>vCU</b>	Virtualized CU
<b>O-DU</b>	O-RAN Distributed Unit
<b>O-CU</b>	O-RAN Centralized Unit
<b>BBU</b>	Baseband Unit
<b>RU</b>	Remote Unit

- O-RU** O-RAN Remote Unit
- LDPC** Low-Density Parity-Check
- FEC** Forward Error Correction
- URLLC** Ultra-Reliable Low-Latency Communication
- eMBB** Enhanced Mobile Broadband
- O-RAN** Open RAN
- COTS** Commercial Off-The-Shelf
- MAC** Medium Access Control
- PHY** Physical
- L-PHY** Low-PHY
- H-PHY** High-PHY
- RRC** Radio Resource Control
- RLC** Radio link Control
- PDCCP** Packet Data Convergence Protocol
- SIMD** Simple-Instruction Multiple-Data
- CPU** Central Processing Unit
- CNI** Container Networking Interface
- NIC** Network Interface Card
- FDD** Frequency Division Duplex
- FR** Frequency Range
- TDD** Time Division Duplex
- LLR** Log-Likelihood Ratio
- RL** Reinforcement Learning
- CB** Contextual Bandit
- AC** Actor-Critic

---

<b>DDPG</b>	Deep Deterministic Policy Gradient
<b>MSE</b>	Mean Squared Error
<b>LTE</b>	Long Term Evolution
<b>CapEx</b>	Capital Expenditure
<b>OpEx</b>	Operating Expense
<b>NR</b>	New Radio
<b>DT</b>	Digital Twin
<b>PT</b>	Physical Twin
<b>CPS</b>	Cyber Physical System
<b>ZSM</b>	Zero-touch Network and Service Management
<b>MR</b>	Machine Reasoning
<b>QPU</b>	Quantum Processing Unit
<b>QA</b>	Quantum Annealer
<b>SA</b>	Simulated Annealer
<b>QUBO</b>	Quadratic Unconstrained Binary Optimization
<b>MM</b>	MinorMiner
<b>DRL</b>	Deep Reinforcement Learning
<b>CQI</b>	Channel Quality Indicator
<b>OAI</b>	Open Air Interface
<b>MIMO</b>	Multiple-Input Multiple-Output
<b>QoS</b>	Quality of Service
<b>RF</b>	Radio Frequency
<b>XAI</b>	Explainable AI
<b>UE</b>	User Equipment
<b>PRB</b>	Physical Resource Block

<b>MCS</b>	Modulation Coding Scheme
<b>CNCF</b>	Cloud Native Computing Foundation
<b>5GC</b>	5G Core
<b>EPC</b>	Evolved Packet Core
<b>CUPS</b>	Control-User Plane Separation
<b>NSA</b>	Non-Standalone
<b>SA</b>	Standalone
<b>eNB</b>	E-UTRAN NodeB
<b>gNB</b>	Next-generation NodeB
<b>SCS</b>	Subcarrier Spacing
<b>NF</b>	Network Function
<b>GRC</b>	GNU Radio Companion
<b>SDR</b>	Software-defined Radio
<b>AWGN</b>	Additive White Gaussian Noise
<b>BPSK</b>	Binary Phase Shift Keying
<b>QPSK</b>	Quadrature Phase Shift Keying
<b>DC</b>	Data Center
<b>NISQ</b>	Noisy Intermediate-Scale Quantum
<b>ICE</b>	Integrated Control Errors
<b>QAM</b>	Quadrature Amplitude Modulation
<b>3GPP</b>	3rd Generation Partnership Project
<b>HARQ</b>	Hybrid Automatic Repeat Request
<b>PDCCH</b>	Physical Downlink Control Channel
<b>PDSCH</b>	Physical Downlink Shared Channel
<b>PUSCH</b>	Physical Uplink Shared Channel

**TB** Transport Block

**UL** Uplink

**DL** Downlink

**TTI** Transmission Time Interval

**BP** Belief Propagation

**BSR** Buffer State Report

**DCI** Downlink Control Information

**SNR** Signal-to-Noise Ratio

**TBS** Transport Block Size

**OFDMA** Orthogonal Frequency-Division Multiple Access

**AI** Artificial Intelligence

**ASIC** Application-Specific Integrated Circuit

**FPGA** Field-Programmable Gate Array

**CRC** Cyclic Redundancy Check

**DL** Deep Learning

**GPU** Graphics Processing Unit

**ML** Machine Learning

**NFV** Network Function Virtualization

**NIC** Network Interface Card

**NN** Neural Network

**SDN** Software-Defined Networking



# 1

## Introduction

---

Over the past decade, the telecommunications industry has entered a new era in which cloud-nativeness is no longer considered an optional advantage, but rather a foundational prerequisite for the successful deployment of 5G and future network services. Cloud-native architectures offer enhanced programmability and dynamic orchestration capabilities, unlocking the potential of advanced service classes such as ultra-reliable low-latency communication (URLLC), as envisioned in 5G standards.

This transformation has been accelerated by the convergence of Software-Defined Networking (SDN) and Network Function Virtualization (NFV). Within this evolving landscape, the Radio Access Network (RAN) – historically a hardware-centric component of mobile systems – has become a strategic focal point for innovation. The emergence of Virtualized Radio Access Network (vRAN) and Open RAN (O-RAN) paradigms aims to disrupt vendor lock-ins, enhance interoperability, and enable flexible, scalable deployment of radio functions on general-purpose Commercial Off-The-Shelf (COTS) hardware. Highlighting this potential, the global vRAN market size was estimated at nearly 16 billion USD in 2024 [6], and according to industry analysts, it is projected to reach 408 billion USD, yielding a compound annual growth rate of 45%. Leading vendors such as Nokia, NEC, Ericsson, NVIDIA, and Qualcomm are actively competing for market share.

The vRAN paradigm has shifted baseband processing from expensive proprietary hardware onto standard cloud infrastructure. This architectural shift involves disaggregating the Baseband Unit (BBU) functions: Low-PHY (L-PHY) layer operations (*e.g.*, FFT/IFFT, beamforming) are offloaded to the Remote Unit (RU); High-PHY (H-PHY) and Medium Access Control (MAC) layer functions are executed by the Virtualized DU (vDU); and higher-layer control functions (*e.g.*, Radio Resource Control (RRC)) and Packet Data Convergence Protocol (PDCP)) are managed by Virtualized CU (vCU). To further promote openness and interoperability, the O-RAN Alliance [7] has introduced standardized interfaces between these functional blocks, facilitating modular development, reducing vendor lock-in, and lowering the total cost of ownership through

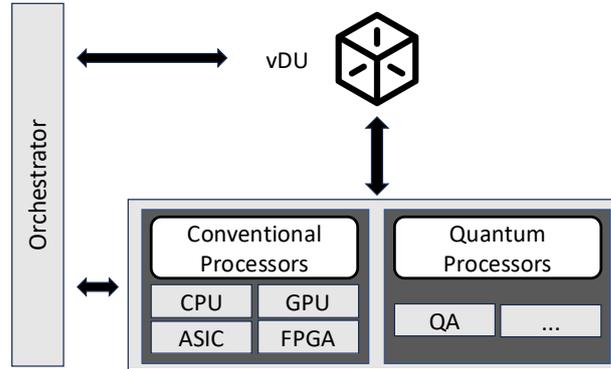


Figure 1.1: vDU hardware resources.

ecosystem diversification.

## 1.1 Motivation

Given the heterogeneity of compute resources available in modern cloud infrastructures, it is imperative to maximize resource efficiency in a manner that is both sustainable and power-conscious. Efficient utilization of these diverse resources ensures that cloud-native deployments, including those supporting vRANs, can meet performance and energy constraints without incurring unnecessary operational costs.

By design, the vDU and vCUs are intended to run in containerized environments atop conventional shared infrastructure, as shown in Fig. 1.1. This architectural choice aims to enable flexible management and orchestration while also reducing the total cost of ownership, as previously discussed. However, the Simple-Instruction Multiple-Data (SIMD)-enabled Central Processing Units (CPUs) alone are insufficient to meet the stringent reliability and latency requirements of 5G services, especially of baseband processing at the Physical (PHY) layer of the vDU [8]. These processors typically fail to satisfy the hard real-time constraints mandated by 5G timelines.

To address this limitation, network operators commonly employ external hardware accelerators, such as Application-Specific Integrated Circuits (ASICs) (*e.g.*, Intel ACC100 [9]) or Field-Programmable Gate Arrays (FPGAs) (*e.g.*, Xilinx T1 [10]), to offload compute-intensive tasks. NVIDIA Graphics Processing Units (GPUs) [11] have taken a rise since they offer a unique advantage: they support concurrent execution of heterogeneous workloads, enabling virtualization of the resources for multiple tenants. This makes it feasible to colocate best-effort Artificial Intelligence (AI) inference jobs alongside latency-sensitive 5G baseband tasks on the same hardware. The left part of Fig. 1.1 lays out the landscape of the conventional hardware resources that can support 5G baseband processing blocks.

Despite the optimistic forecasts for 5G vRAN revenue streams, the real-world adoption has been more cautious. Due to the densification of the network, the initial Capital Expenditure (CapEx) rose since around 7x more fiber deployment is needed compared to Long Term Evolution (LTE) networks, as well as massive Multiple-Input Multiple-Output (MIMO) and specialized antennas require more expensive hardware [12]. Additionally, surveys indicate that only 10% of the total RAN baseband product market was attributed to vRAN in 2024, with expectations that this figure will not exceed 20% before 2028 [13].

This can be attributed to several factors. On the one hand, while hardware accelerators significantly accelerate baseband processing, they introduce additional layers of complexity. They deviate from the core vision of vRAN, which emphasizes flexibility, manageability, and infrastructure sharing. These devices are often rigid in their orchestration and difficult to virtualize across multiple tenants or workloads, thus contributing to lower resource efficiency. On the other hand, despite GPU-based vRANs supporting multiple tenants, their widespread deployment is hindered by significant capital expenditures. The initial investment required to outfit edge sites with vRAN-capable GPUs is substantial [14]. The vision of monetizing idle GPU capacity by hosting third-party AI workloads and generating additional revenue streams [15] remains theoretical and has yet to demonstrate sustainable business viability. Furthermore, GPUs consume considerably more power than CPUs, raising concerns over operational efficiency and energy costs [8]. These challenges underscore the need for sustainable COTS infrastructure that can enable resource sharing across multiple tenants and efficiently support heterogeneous workloads alongside 5G. They also highlight the importance of exploring alternative technologies that offer improved resource efficiency, particularly in terms of reduced power consumption.

For vRAN to be economically sustainable, it must be capable of running efficiently on CPU-only COTS infrastructure. This would enable seamless resource sharing with third-party applications and reduce management overhead. Achieving this requires enhancements to the vDU software stack, enabling it to monitor and react to host compute load dynamically. In particular, the vDU must incorporate compute-aware logic that adjusts its internal behavior based on the state of the shared infrastructure, ensuring both reliability and coexistence with non-RAN workloads.

To support such compute-aware behavior, it becomes essential to incorporate intelligent control mechanisms within the vDU. Given the dynamic workload patterns and fluctuating resource availability in data center environments, the deployment of advanced algorithms based on AI and Machine Learning (ML) is critical. These algorithms can facilitate real-time resource management and adaptive behavior of the vDU. However, effective training and deployment of ML models necessitate an accurate and structured representation of the vDU stack. This is where the concept of Digital Twin (DT) becomes

valuable. As a virtual replica of the vDU, a digital twin enables offline experimentation, optimization, and continuous refinement of ML strategies. Its ability to mirror real-time system dynamics makes it particularly well-suited for mobile network environments, where reliability and responsiveness are paramount.

Orthogonal to the CPU-GPU-hardware accelerators spectrum, future mobile network stacks must also consider alternative paradigms to meet the goals of high reliability and low energy footprint, leading to an overall more sustainable infrastructure. Motivated by this need, the quest is open for more capable and power-efficient accelerators, and one emerging technology stands out as a promising candidate: *Quantum computing*. Quantum computing exhibits extraordinary potential in solving high-dimensional, non-convex optimization problems, inherently encountered in wireless communications. This makes quantum-based solutions particularly attractive for RAN baseband processing. Quantum Processing Units (QPUs) with a sufficiently large number of qubits hold the promise of meeting the strict timing constraints and reliability demands of tasks such as Forward Error Correction (FEC) decoding [16], even in computationally intensive scenarios like massive MIMO, all while minimizing power consumption and operational costs.

Seminal studies have demonstrated that quantum systems can approach or match the performance of conventional hardware while offering orders-of-magnitude improvements in energy efficiency, thereby significantly reducing operational expenditures [17–19]. In this direction, quantum machines should be investigated as possible alternatives in the accelerator pool of the cloud infrastructure. This envision is illustrated in the right part of Fig. 1.1, where quantum resources, such as Quantum Annealers (QAs) [20], are placed alongside the conventional resources, ready to support the real-time functionality of vDU.

## 1.2 Research Challenges

The transition towards virtualized, easy-to-manage low-energy footprint RAN infrastructures with optimized resource allocation introduces a wide array of research challenges that span the architectural, algorithmic, and computational domains. While the vRAN paradigm promises flexibility, scalability, and lower operational costs, its implementation requires rigorous and thorough investigation. These challenges can be broadly categorized as follows:

### 1. Efficient Baseband Processing on Shared Infrastructure

Achieving high-throughput, low-latency baseband processing on general-purpose CPUs remains a non-trivial task. Existing acceleration strategies offer performance benefits but at the cost of rigid orchestration [11], and higher power consumption [8]. By contrast, CPU-only vRAN deployments represent a more sustainable solution in terms of

orchestration, power and cost: CPUs integrate seamlessly with orchestration frameworks (unlike ASICs and FPGAs), reducing operational complexity, while also consuming less power and incurring lower cost compared to specialized accelerators such as GPUs. Making such deployments viable, particularly at the network edge, requires software-based techniques capable of extracting maximum efficiency, through compute-aware algorithms that gracefully handle workload variability and infrastructure sharing.

Traditional scheduling algorithms often overlook the dynamic interplay between wireless link conditions and computational resource availability. Designing intelligent scheduling policies that can learn and adapt to real-time context-spanning both radio and compute metrics, is critical to optimizing Quality of Service (QoS) while ensuring reliability.

To effectively evaluate such policies, it is essential to develop a cloud-native 5G emulation framework that reflects realistic deployment conditions. A key barrier to that is the lack of open-source and affordable platforms that can support experimentation under multi-user and compute-constrained scenarios. Existing frameworks are often proprietary, cost-prohibitive, or tailored to vendor-specific architectures, limiting their accessibility and reproducibility within the research community. This thesis directly addresses this gap by designing and implementing an open, cloud-native 5G emulation framework that enables experimentation under multi-user and compute-constrained scenarios. This framework provides a standardized reference architecture for investigating the interaction between radio conditions and computational resource availability, thereby supporting the development and evaluation of sustainable, CPU-only vRAN solutions.

## 2. Integration of Quantum Computers in the RAN

As the limitations of conventional accelerators become more pronounced, there is a growing interest in leveraging emerging computing paradigms such as quantum computing. Quantum computers could partially replace certain function blocks in the physical layer pipeline of the base stations, due to significant energy footprint advantages they provide compared to their traditional computing counterparts, as recent studies highlight [17], thus leading to better energy sustainability. However, integrating quantum co-processors into RAN architectures raises novel challenges in terms of algorithm design, system integration, and real-time responsiveness. Mapping wireless signal processing problems—such as FEC decoding—onto QPUs while ensuring that execution meets telecom-grade latency and reliability constraints is an open area of research.

This thesis attempts to address the above challenges by developing intelligent, context-aware orchestration mechanisms for vRAN, leveraging AI/ML and digital twin paradigms, and by investigating the feasibility of quantum-assisted baseband processing.

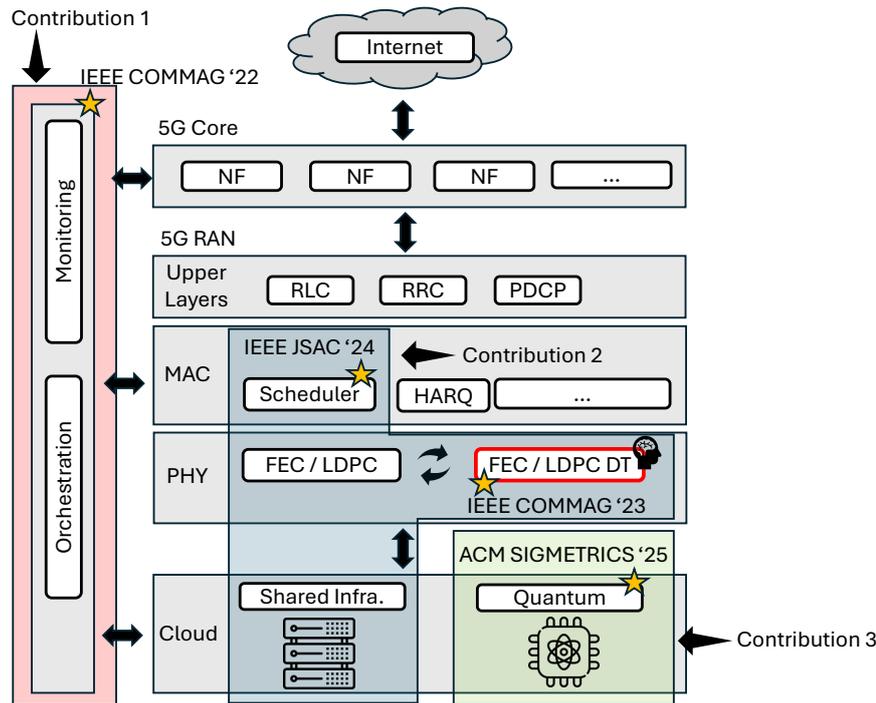


Figure 1.2: Thesis contributions.

### 1.3 Contributions

This thesis has yielded several key contributions in three areas: end-to-end 5G network orchestration, CPU-aware vDUs, and quantum-enhanced 5G RAN. In the domain of 5G network orchestration, the proposed architecture has been published in *IEEE Communications Magazine (COMMAG)* (2022). In the domain of CPU-aware vDUs, the research has resulted in two peer-reviewed journal publications—one in *IEEE Journal on Selected Areas in Communications (JSAC)* (2024) and one in *IEEE COMMAG* (2023). Notably, both *IEEE JSAC* and *IEEE COMMAG* are ranked in Quartile 1 (Q1) according to the Clarivate Analytics Journal Citation Reports (JCR) and have Journal Impact Factor (JIF) 17.2 and 8.2, respectively. In the area of quantum-based 5G RAN, the findings were presented at *ACM SIGMETRICS* (2025), a venue recognized in the CSrankings database and rated as an A\* conference by the CORE2023 classification. All three contributions are illustrated in the 5G network landscape of Fig. 1.2.

The contributions of this thesis are summarized as follows:

#### **Contribution 1.** *Cloud-Native RAN and Core Design*

The first contribution of this thesis is the design and deployment of a fully cloud-native 5G Core (5GC) and RAN on COTS hardware using open-source software, eliminating the need for specialized Radio Frequency (RF) front-end hardware. The complete source code

has been made publicly available on GitHub. To the best of the author's knowledge, this represented the first open implementation of such a design at the time this work was carried out, democratizing access to cloud-native wireless research for practitioners by removing dependencies on expensive proprietary hardware and vendor-specific solutions. The key elements of this work include containerization of 5GC Network Functions (NFs) and their orchestration within a Kubernetes cluster, and performance evaluation under various load and mobility scenarios using RAN front-end emulation without requiring physical RF components. This cloud-native testbed was fully detailed in the following publication.

- **Nikolaos Apostolakis**, Marco Gramaglia and Pablo Serrano, "Design and Validation of an Open Source Cloud Native Mobile Network," in *IEEE Communications Magazine*, vol. 60, no. 11, pp. 66-72, November 2022, doi: 10.1109/MCOM.003.2200195

**Contribution 2.** *CPU-Aware MAC Scheduling*

The second contribution is the design and implementation of a CPU-aware MAC scheduler, named ATHENA, which addresses the *noisy neighbor* effect commonly encountered in cloud data center environments. ATHENA is an ML-based resource controller that dynamically adjusts the allocation of uplink resources to users. By doing so, it directly influences the timing of the uplink processing pipeline and ensures that critical latency deadlines are not violated. ATHENA was evaluated against a conventional baseline scheduler and demonstrated superior performance. To improve the interpretability of the scheduling decisions, this work also integrated Explainable AI (XAI) techniques in order to offer insights into the decision-making process of the ML model.

Furthermore, a digital twin of the physical layer was developed to emulate the processing environment. This enabled more efficient and repeatable iterations during model training and validation, reducing the ML development and evaluation lifecycle. The development and deployment of ATHENA were built upon the cloud-native testbed established in **Contribution 1**. The two publications that came out of this work are listed below.

- **Nikolaos Apostolakis**, Marco Gramaglia, Livia Elena Chatzieftheriou, Tejas Subramanya, Albert Banchs and Henning Sanneck, "ATHENA: Machine Learning and Reasoning for Radio Resources Scheduling in vRAN Systems," in *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 2, pp. 263-279, Feb. 2024, doi: 10.1109/JSAC.2023.3336155

- **Nikolaos Apostolakis**, Livia Elena Chatzieftheriou, Dario Bega, Marco Gramaglia and Albert Banchs, "Digital Twins for Next-Generation Mobile Networks:

Applications and Solutions," in *IEEE Communications Magazine*, vol. 61, no. 11, pp. 80-86, November 2023, doi: 10.1109/MCOM.001.2200854

**Contribution 3.** *Quantum-based FEC Processors*

This work introduces **Qu4Fec**, a novel solution for designing and implementing a FEC decoder—specifically for Low-Density Parity-Check (LDPC) codes—using quantum annealers. **Qu4Fec** presents a new formulation of the LDPC decoding problem that aligns with established code design principles and highlights their importance in the context of quantum-assisted decoding. The experiments showed that through simulated annealing, **Qu4Fec** achieved significant performance gains over the state-of-the-art. Moreover, this contribution identifies two critical limitations of current quantum hardware: *i*) non-ideal embeddings and *ii*) scaling and quantization effects, which help explain the noisy results observed when deploying the decoder on the actual quantum machines, that prevent them from being used in production-grade communication systems. Although current quantum annealers cannot realistically support vRAN operations in the near future, this contribution ultimately suggests hardware extensions to qubit layouts that could reduce qubit utilization and noise. Such improvements, while longer-term, would move quantum hardware closer to partially supporting LDPC decoding in practical systems.

- **Nikolaos Apostolakis**, Marta Sierra-Obea, Marco Gramaglia, Jose A. Ayala-Romero, Andres Garcia-Saavedra, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. 2025. Quantum Computing in the RAN with Qu4Fec: Closing Gaps Towards Quantum-based FEC Processors. *Proc. ACM Meas. Anal. Comput. Syst.* 9, 2, Article 36 (June 2025), 25 pages. <https://doi.org/10.1145/3727128>

- **Nikolaos Apostolakis**, Marta Sierra-Obea, Marco Gramaglia, Jose A. Ayala-Romero, Andres Garcia-Saavedra, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. 2025. Quantum Computing in the RAN with Qu4Fec: Closing Gaps Towards Quantum-based FEC Processors. In *Abstracts of the 2025 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '25)*. Association for Computing Machinery, New York, NY, USA, 64-66. doi: 10.1145/3726854.3727311

The publicly available artifacts of the contributions above are listed in Table 1.1.

	URL	Paper
Cloud-Native 5G Core	<a href="https://github.com/kaposnick/k8s-open5gs">github.com/kaposnick/k8s-open5gs</a>	[1]
ATHENA	<a href="https://github.com/kaposnick/athena_agent">github.com/kaposnick/athena_agent</a>	[2]

Table 1.1: Software released as open source.

## 1.4 Outline

The following chapters delve deeper into the technical contributions of this thesis. Table 1.2 provides a visual overview of the relationships among research challenges, contributions, and chapters.

Research Challenges	Contributions	Chapters	Papers
Efficient Baseband Processing on Shared Infrastructure	Cloud-Native RAN and Core Design	2,3	[1]
	CPU-Aware MAC Scheduling	2,4,5	[2, 3]
Integration of Quantum Computers in the RAN	Quantum-based FEC Processors	2,6	[4, 5]

Table 1.2: Mapping of research challenges to contributions, chapters and papers.

Chapter 2, the background chapter, provides a brief introduction to 5G New Radio (NR) and outlines the LDPC encoding and decoding processes in the context of vRANs. It discusses the critical role of efficient MAC scheduling in enhancing resource utilization, followed by an exploration of how AI/ML can be leveraged to optimize resource allocation and provide explainability in decision-making. The chapter also highlights the potential of DT technology in 5G, particularly in redefining the way networking algorithms are developed and deployed. Finally, it concludes by presenting state-of-the-art works in MAC scheduling, advancements in CPU-aware RANs, and quantum-based baseband processing.

Chapter 3 presents the cloud-native 5GC and RAN architecture developed using open-source technologies. It details the end-to-end design, beginning with Management and Orchestration, followed by RAN monitoring and RF front-end emulation, enabling the execution of complex load and mobility scenarios. The chapter concludes with a discussion of the evaluation setup, including various RAN mobility scenarios and the dynamic allocation of user-plane function instances in the core network.

Chapter 4 builds upon the testbed presented in the previous chapter and introduces **ATHENA**, an ML-based, CPU-aware MAC radio resource controller designed to mitigate deadline violations in shared compute environments. The chapter discusses the noisy neighbor problem and its consequences, which arise in virtualized infrastructure where co-located workloads contend for shared CPU resources, necessitating dynamic AI/ML-based resource management strategies. **ATHENA** intelligently allocates uplink radio resources to users, coordinating resource distribution to avoid over-provisioning and ensuring that critical processing deadlines are met.

Chapter 5 explores complementary aspects and design decisions associated with **ATHENA**. Specifically, it introduces three explainability techniques that provide transparency into the internal workings of **ATHENA**'s ML model, fostering trust and interpretability for network operators. Additionally, the chapter discusses the development of a digital twin for the LDPC decoder, highlighting its role in accelerating and facilitating the training, evaluation, and iterative refinement of the ML model behind

ATHENA.

Chapter 6 introduces `Qu4Fec`, a solution for implementing an LDPC decoder on quantum annealers. This chapter highlights the critical role of established code design best practices and their impact on the error-correcting capabilities of the decoder. It also presents a new problem formulation for LDPC decoding tailored to quantum annealing, identifies current hardware limitations that constrain the practical use of quantum computers in this domain, and proposes potential hardware extensions aimed at mitigating these challenges.

Finally, Chapter 7 summarizes the key findings of this thesis and outlines potential directions for future research inspired by the presented thesis.

# 2

## Background

---

This chapter provides the technical background to understand the main contributions of this thesis. In Sec. 2.1, I show an overview of the 5G NR protocol stack as long as the uplink resource allocation procedure, before giving a primer to LDPC codes (Sec. 2.1.1), essential for decoding in 5G systems and their impact on vRANs. In Sec. 2.1.2, there is a brief explanation of the MAC scheduler, while in Sec. 2.1.3 and Sec. 2.1.4, I position digital twin and XAI, respectively, with regards to network operators and 5G.

In Sec. 2.2.1, I introduce the concepts of QAs and their importance for RAN consolidation (Sec. 2.2.2). Last, in Sec. 2.3, I display the state-of-the-art from the literature, ranging from ML MAC scheduling and CPU-aware RAN to mobile networking software solutions and Quantum baseband processing advancements.

### 2.1 5G NR

The major focus of this thesis is on the PHY and MAC layers of the 3rd Generation Partnership Project (3GPP) 5G NR stack [21] of the 5G base station, namely Next-generation NodeB (gNB). These layers take care of several functionalities, ranging from very L-PHY level functionality, *e.g.*, (de-)modulation and encoding/decoding, to radio resource scheduling to User Equipments (UEs). These layers, up to Radio link Control (RLC), are part of the O-RAN Distributed Unit (O-DU). O-DU is complemented by the O-RAN Remote Unit (O-RU) and O-RAN Centralized Unit (O-CU) that provide low-level baseband processing (*e.g.*, FFT/IFFT) and higher-level (*e.g.*, IP segmentation/concatenation) services to the O-DU, respectively.

The gNB manages the available amount of bandwidth. 3GPP has designated two Frequency Ranges (FRs), namely FR1 and FR2, dedicated for sub-6 GHz and above 6 GHz deployments respectively. FR1's bandwidth ranges from 5 to 100 MHz, while FR2's range is between 50 and 400 MHz. gNB's PHY and MAC layer make decisions and process data on every *slot*, that is the minimum time unit of operation. Slot duration in 5G is  $2^{-\mu}$  ms, and varies depending on the numerology used  $\mu \in \{0, 1, 2\}$  for FR1 and  $\mu \in \{3, 4\}$

for FR2. Each slot occupies up to 12 or 14 symbols with a duration of  $2^{-\mu} \cdot 66.67 \mu\text{s}$ .  $\mu$  slots comprise a *subframe* and 10 subframes comprise a *system frame*.

Depending on the available bandwidth and numerology, there is a maximum number of Physical Resource Blocks (PRBs). One PRB encompasses 12 subcarriers of bandwidth  $2^{\mu} \cdot 15$  KHz each. 5G NR uses the Orthogonal Frequency-Division Multiple Access (OFDMA) scheme for data transmission, encoding data allocated to a UE into a portion of these PRBs.

The Modulation Coding Scheme (MCS) index captures the number of bits that will be encoded in each transmitted symbol as well as the coding rate. The coding rate affects the number of parity bits, extra bits appended to the original message in order to decode the erroneous bits at the receiver side. For 5G NR, 3GPP defines a set of MCS indexes, ranging from the lowest QPSK capable of carrying 2 bits/symbol, up to 256-QAM that can carry 8 bits/symbol. Combined with MIMO techniques, these MCSs provide the Gbps data rates obtained by 5G. The MCS that will be used by the UEs depends on the estimated channel conditions, which are upper bounded by Shannon's law.

Overall, the system consists of gNBs that provide communication capabilities to UEs. The UEs communicate with the gNB to transmit their information, according to the radio resource scheduling policies defined by the gNB. These policies are effectively translated into a set of transmission parameters, such as the number of PRBs, MCS, number of transmission layers *etc.*

According to the 3GPP 5G NR specification [22], each Uplink (UL) frame transmission is associated with one Hybrid Automatic Repeat Request (HARQ) process, which can handle up to two transmissions at a time. When the UE requests to send data, the gNB assigns a HARQ process as responsible for managing the whole data transmission lifecycle. The gNB issues a scheduling *grant* using Downlink Control Information (DCI) in the Physical Downlink Control Channel (PDCCH). After the UL frame reception within the Physical Uplink Shared Channel (PUSCH) by the L-PHY, a PHY-thread starts processing and directs it to the H-PHY layer containing several processing modules, *i.e.*, channel estimator, equalizer, demodulator, decoder *etc.* In Sec. 2.1.1.3, the role of the PHY-thread is further described. The H-PHY will start sequentially decoding the UL frame of each user and will inform the corresponding HARQ process about the ACK/NACK of the UL-transmitted Transport Block (TB). In the case of ACK, the UE can use this HARQ process for a new transmission. In the case of NACK, the HARQ process will direct the UE to retransmit the TB using a different *redundancy version*, *i.e.*, a different set of systematic and redundant bits, that will be combined with the previous transmission to increase the probability of successful decoding.

However, all the tasks that are related to UL frame processing (at the PHY layer) have to be completed before the DL frame, containing the ACK/NACK, is scheduled to be sent. The ACK/NACK is contained in the PDCCH, located in the first symbols of the

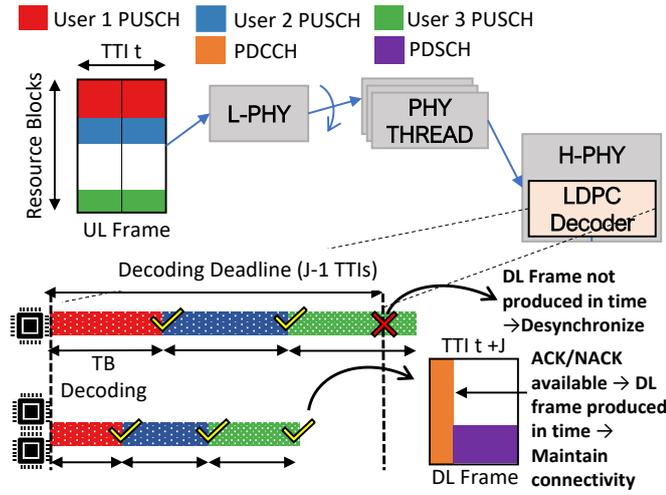


Figure 2.1: Uplink Processing Pipeline in 5G NR. The received UL frame is directed to the H-PHY, which processes the users' TBs. *i*) Low capacity/highly congested CPUs: Total processing time exceeds the deadline, and the ACK/NACK is not integrated into the DL frame in time, incurring users' desynchronization. *ii*) High capacity/low congested CPUs: Processing tasks end before the deadline, and the information regarding ACK/NACK is packed in the PDCCH of the DL Frame, maintaining connectivity.

DL frame, while the rest symbols transmit the downlink data in the Physical Downlink Shared Channel (PDSCH). While in 4G ACK/NACK had a dedicated field, it is implicit in 5G – it comes via scheduling grants within DCI fields of PDCCH. This introduces a hard deadline for the completion of the processing tasks (that are by far the most computationally expensive of the full protocol stack [23–26]). In 5G NR, this deadline is configurable and let it be equal to  $J - 1$  slots. That is, the frame processing result has to be ready before the ACK/NACK transmission that will take place in the  $J$ -th Transmission Time Interval (TTI) after the reception of the original frame. In 5G NR, the value  $J$  is programmable, but the default values for the TTI length (*i.e.*, 1 ms) and  $J$  (*i.e.*, 4 ms [8]) impose a deadline of 3 ms to complete demodulation, rate dematching, and decoding operations [23]. In Fig. 2.1, the UL frame processing procedure is depicted under two scenarios; *i*) CPUs that are congested with third-party external tasks (also referred to as low capacity CPUs), for which processing deadline violations, and thus user desynchronization, occur and *ii*) uncongested CPUs (*i.e.*, with high available capacity), where the frame can be processed within the deadline, and the ACK/NACK can be transmitted within the deadline.

A deadline violation has critical consequences on the vRAN operation, with sudden drops in the achieved performance [25], and it must be avoided by properly configuring the required computing resources. However, this is not an easy task in vRANs, where computing resources are dynamically orchestrated. Only with very high overprovisioning, which may lead to severely unsustainable systems [27], statistical guarantees of avoiding

computing resource shortages are obtained.

This is needed because the usage of shared, cloud computing platforms introduces the so-called *noisy neighbor* problem [28]: Let different processes, *e.g.*, different parts of the implementation of a vRAN system running on the same platform, share a computing platform. In this case, there may be capacity oscillations due to competing processes, which may lead the elapsed time for the completion of the task to exceed the target deadline.

Observe that vRANs are extremely fragile systems when dealing with computing capacity oscillation: Missing the deadline, even for just milliseconds, can cause the UE to completely lose synchronization, and hence the associated data flow [25]. This would cause additional congestion, as the same packets would need to be sent again. One of the most critical blocks in the uplink processing pipeline is the LDPC decoder [8, 25, 29], whose overview is given below.

### 2.1.1 LDPC Codes

An LDPC [30] code is a binary linear FEC code that is characterized by a sparse generator matrix  $G \in \{0, 1\}^{k \times n}$  and parity-check matrix  $H \in \{0, 1\}^{m \times n}$ , where  $k$ ,  $n$  is the number of message and codeword bits respectively, and  $m$  is the row count of  $H$ . A codeword  $\hat{c}$  is valid if and only if it satisfies  $H \cdot \hat{c}^T = 0$  in the binary domain, or equivalently,  $h_i \cdot \hat{c}^T = 0$ , also known as check constraints, where  $h_i$  with  $i=1, \dots, m$  is the row vector  $i$  of  $H$ . A  $(d_v, d_c)$ -regular LDPC code involves exactly  $d_c$  codeword bits in each constraint, while each codeword bit is present in  $d_v$  constraints. I denote by  $(d_v, d_c, n)$  the  $(d_v, d_c)$ -regular LDPC code of length  $n$ . An LDPC code can be visually represented by the Tanner graph [31], a bipartite graph consisting of two sets of nodes: check nodes and variable nodes. Check nodes are the rows of the parity-check matrix, *i.e.*, match the constraints, while the variable nodes represent columns, *i.e.*, the codeword bits.

#### 2.1.1.1 Encoding

The code design eventually yields a parity check matrix  $H$ , with properties that assure the code's error correction capabilities. Converting the matrix  $H$  into a generator matrix  $G$  assumes that  $H$  is a full-rank matrix, and if not, Gaussian elimination method needs to be performed first.

1. **Convert  $H$  into a systematic form:**  $H = [P|I_m]$ , where  $P$  is an  $m \times (n - m)$  matrix and  $I_m$  is the  $m \times m$  identity matrix.
2. **Construct the generator Matix:**  $G = [I_k|P^T]$ , where  $P^T$  is the transpose of the matrix  $P$ .

Given an original message  $a$  of  $k$  bits and the generator matrix  $G$ , a codeword is given as  $c = aG$ . The codewords generated by  $G$  satisfy the parity check conditions, and it must hold  $HG^T = 0$ , confirming  $G$  generates valid codewords for  $H$ .

### 2.1.1.2 Decoding

The LDPC maximum likelihood (ML) decoder outputs the codeword  $x \in \mathcal{C}$  that maximizes the a-posteriori probability  $P(x|y)$ , where  $\mathcal{C}$  is the valid codeword set and  $y$  is the received channel value vector. Expressing this mathematically, it is

$$\hat{x} = \arg \max_{x \in \mathcal{C}} P(x|y). \quad (2.1)$$

Since the cardinality of  $\mathcal{C}$  grows exponentially with  $k$ , exhaustive search is not viable for practical applications. Instead, different efficient algorithms have been proposed to solve (2.1).

Belief Propagation (BP) [32] is a heuristic algorithm that attempts to solve the decoding problem via iterative message passing between the check and variable nodes of the Tanner graph. The messages improve the extrinsic information that is received in one variable node from the rest of the variable nodes, which eventually strengthens the belief of a certain bit being 0 or 1. The input of the algorithm is the Log-Likelihood Ratio (LLR) vector:  $LLR_i = \log \frac{Pr(b_i=0|y)}{Pr(b_i=1|y)}$ , where  $Pr(b_i = k|y)$ ,  $k \in \{0, 1\}$ ,  $i = 0..n - 1$  is the probability for bit  $b_i = k$  given received vector  $y$ . The calculation of  $LLR_i$  depends on the modulation scheme used and the noise variance of the channel. The algorithm ends when  $H \cdot \hat{c}^T = 0$  or a maximum number of iterations is reached.

I now describe the BP algorithm. Let the set of check nodes connected to bit node  $n$  as  $\mathcal{M}(n)$  and the set of bit nodes connected to check node  $m$  as  $\mathcal{N}(m)$ . Also, denote the message from check node  $m$  to variable node  $n$  as  $Z_{mn}$  and the variable  $n$  to check node  $m$  message as  $L_{nm}$ . The BP decoding algorithm comprises the following steps [32]:

1. **Check to variable message update:** For each  $m$ :

$$L_{nm} = 2 \tanh^{-1} \left( \prod_{n' \in \mathcal{N}(m) \setminus n} \left( \tanh \left( \frac{Z_{mn'}}{2} \right) \right) \right)$$

2. **Variable to check message update:** For each  $n$ :

$$Z_{mn} = LLR_n + \sum_{m' \in \mathcal{M}(n) \setminus m} (L_{nm'})$$

3. **Belief update** For each  $n$ :

$$LLR_n := LLR_n + \sum_{m' \in \mathcal{M}(n)} (L_{nm'})$$

4. **Hard decision:** The candidate codeword  $\hat{c} = [\hat{c}_0, \hat{c}_1, \dots, \hat{c}_{n-1}]$  is derived, setting  $\hat{c}_i = 1$  if  $LLR_i \leq 0$  else 0. The decoding exits if  $H \cdot \hat{c}^T = 0$  or a maximum number of iterations is reached. If not, the algorithm proceeds to the next iteration in Step 1.

Different variations of BP have been studied to simplify the computationally expensive Step 1, *e.g.*, Min-sum BP [33], or to speed up the algorithm’s convergence, *e.g.*, Layered BP [34].

### 2.1.1.3 Decoding in vRAN

In a 3GPP PHY pipeline, a new decoding task takes place at every slot where UL user transmissions are instructed. Decoding tasks translate the modulated and encoded digital symbols that are received in the PUSCH, which carries UL data, into bytes associated with each UE that are delivered to the higher layer of the gNB stack.

After demodulating the symbols and performing other channel equalization procedures, each bit belonging to the original transmitted codeword is decoded using an iterative procedure. During this process, each bit  $i$  is represented as a value  $LLR_i$  in the range  $(-\infty \dots +\infty)$ , carrying the log-likelihood ratio of the  $i$ -th bit being either a 0 or a 1. As described in Sec. 2.1.1, the decoder algorithm loops across all the  $LLR$  vector items  $LLR_i$  until the parity check is successful or a maximum number of iterations is reached.

An LDPC decoder instance is present in every PHY layer thread (namely, DSP worker in Fig. 2.1) that handles UL frame processing in each slot. Since the entire UL processing pipeline can typically take longer than a single slot, multiple threads are deployed to ensure that UL frames in consecutive TTIs can be served without delay. As a point of reference, the open-source 4G/5G RAN implementation `srsRAN` [35] deploys 3 PHY-layer threads by default. If an UL frame is received and all threads are occupied, the UL pipeline pauses until a thread becomes available. Furthermore, if multiple users are scheduled in a slot, the decoder of the respective PHY thread processes their transport block sequentially, after which the UL pipeline resumes. As a result, if the decoding time exceeds the deadline, the connected users will become desynchronized from the network and experience a degradation in the quality of service.

### 2.1.2 MAC Scheduler

The *Radio Resource Scheduler* is a component of the MAC layer in the O-DU that performs, among other tasks, the following three functions: *i*) selecting which users are allowed to transmit, *ii*) allocating PRBs, and *iii*) choosing the MCS for the selected users. The UL radio resource scheduling is performed by a centralized agent in the gNB, *e.g.*, the network operator, and consists of deciding the user granted data for the UL channel at every slot, or TTI. Periodically, each user sends a Buffer State Report (BSR) to the gNB containing the user’s UL demand, *i.e.*, the size (in bytes) of the total data pending in the UE’s stack. Then, the gNB schedules the user transmission and answers with a *uplink transmission grant*. The grant DCI contains many fields, among which are: (*i*)

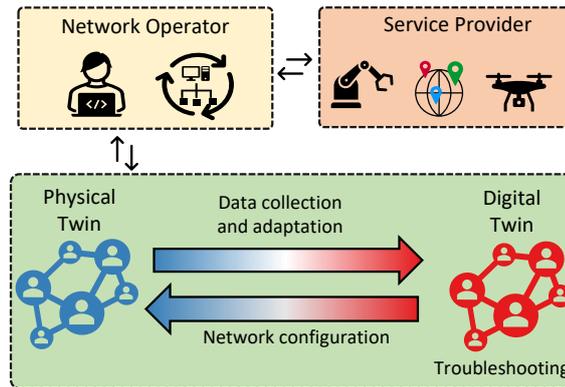


Figure 2.2: Interaction of Physical and Digital Twins in a service provider context.

the number of PRBs, and (ii) the MCS index that the user must use. The choice of the granted user and the selection of PRBs and MCS depend on the specific controller algorithm implemented by the gNB, which uses information such as the user's data size or their channel conditions, *e.g.*, their Signal-to-Noise Ratio (SNR), that would limit the communication.

The selection of MCS and PRBs eventually leads to a given amount of data that can be transmitted by a UE, *i.e.*, TB, whose Transport Block Size (TBS), accounts for the user data, the size of the extra information such as Cyclic Redundancy Check (CRC), and filler bits. Given the discrete number of PRBs and MCS, the possible values of TBS can be found using a simple deterministic procedure [36].

### 2.1.3 Digital Twin in 5G

Digital Twins are expected to increase operational efficiency due to their ability to create virtual representations of real physical objects and processes, known as Cyber Physical Systems (CPSs). CPSs consist [37, 38] of (i) a *physical space* that captures the physical object or entity, also called Physical Twin (PT), (ii) a *virtual space* that captures a cyber representation of the physical twin, also called Digital Twin (DT), and (iii) a *link* that is used for communication between the physical and the virtual spaces. This link enables updates to the virtual model when a change occurs in the real object. This link between the real and the virtual space constitutes the main difference between CPSs and traditional models for predicting entities' behavior in the physical world.

The use of DTs is a key technology for 5G and beyond mobile networks, as they will revolutionize the way network automation is performed. Fig. 2.2 depicts the integration of DTs in a communications provider setting, where DTs coexist with physical objects and reflect their behavior by continuous adaptation to new data. DTs can directly apply network reconfigurations or suggest new ones to the network operator. Similarly, this approach can be applied to the interface between network operators and service providers,

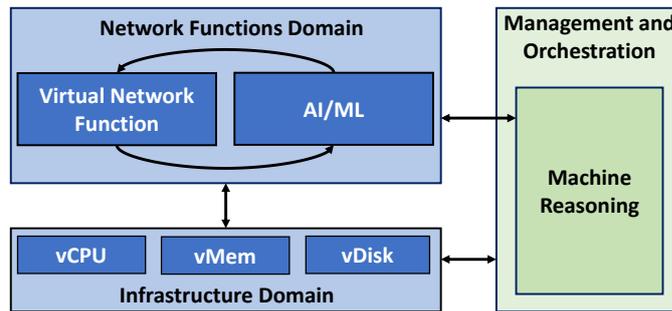


Figure 2.3: Machine Reasoning in network management and orchestration.

allowing the latter to test a service without even having to field-trial it.

DTs can use a variety of methods, from traditional statistical and analytical models to data-driven techniques such as machine learning algorithms. Compared to traditional network simulators, DTs have the advantage of being linked and real-time synchronized with the physical twins they represent. ML techniques can leverage this link to accurately model the physical object’s behavior by observing its historical and current data. The use of such methods to build DTs will result in the development of native-AI 6G networks, which can iterate faster and automatically adapt to changes in the underlying physical system by substituting digital with physical components.

When the real system faces states that it had not encountered before, a *distribution shift* takes place. In such a case, DTs leverage data provided by the PTs and adjust their internal model accordingly. This procedure relies on the link between the PT and the DT, called *data adaptation*. Along with real-time *data collection*, data adaptation captures the actual mobile network status and accurately models the physical twin’s behavior. When the DT’s internal model captures the physical twin with enough accuracy, the DT may request a resource *reconfiguration* at the physical twin to make it operate more efficiently.

#### 2.1.4 AI in 5G

For the zero-touch networking, promoted by the Zero-touch Network and Service Management (ZSM) [39], it is critical that networking components across different domains (*e.g.*, RAN, 5GC, transport) operate autonomously. This integration will create control loops, harvesting huge amounts of data, and intelligent algorithms orchestrating the networks without human supervision. The presence of AI/ML is considered imperative in order to efficiently manage the mobile network’s critical resources. Such application of AI in mobile networks is the radio resource scheduling at the MAC level, where AI algorithms replace the traditional control algorithms (*e.g.*, round-robin, proportional fair). This extra layer introduces additional complexity and new issues, such as the *explainability* of the decisions made.

The introduction of explainable intelligence [40] has helped to demystify the

traditionally opaque, closed-box nature of AI/ML models, enabling network operators to better interpret and understand the rationale behind algorithmic decisions [41]. Building on this foundation, the concept of *Machine Reasoning (MR)* has recently emerged [42], aiming to emulate human analytical reasoning by generating decisions based on the outcomes produced by intelligent algorithms.

Although the idea of reasoning atop intelligent systems dates back years [43], recent advancements in computing have made it feasible to implement such reasoning techniques in practice for managing network intelligence. An illustrative example of MR applied within the mobile networking context is shown in Fig. 2.3. Here, by interpreting the outputs of an AI/ML module, an MR component embedded within the network orchestrator can infer the underlying causes of specific decisions—such as resource scarcity—and subsequently trigger additional actions, such as re-orchestration, to adapt the system accordingly.

## 2.2 Quantum Computing

Quantum computers are emerging as an alternative baseband signal processing platform because of their ability to solve complex combinatorial problems, inherently existing in the lower layers of the mobile networking stack (*e.g.*, MIMO, FEC, channel estimation). Followingly, there is a brief introduction to QAs before analyzing the power benefits of running certain mobile station tasks on Quantum computers.

### 2.2.1 Quantum Annealers

Quantum annealing [44] is a Quantum computing approach designed to solve optimization problems by finding the global minimum of a given function. It harnesses Quantum phenomena occurring in specialized superconducting loops at extremely low temperatures near absolute zero, including qubit superposition, tunneling, and entanglement. This process is carried out by a Quantum annealing machine, known as Quantum Annealer (QA), which shares conceptual similarities with the Simulated Annealer (SA), a metaheuristic that can be coded and executed on a classical processor. The QA system is configured such that its lowest energy state coincides with the solution to the problem under consideration. Both techniques start with a high-energy state and seek lower-energy states by cooling the system. During the high-temperature annealing phase, the system explores various possible solutions to avoid getting stuck in local minima. As temperature decreases, the system moves to lower energy states, lowering the probability of escaping them and ultimately seeking the global optimum.

Quantum annealing, like simulated annealing [45], is suitable for solving objective functions formulated as Quadratic Unconstrained Binary Optimization (QUBO) problems [46]. A QUBO model aims to find the vector of binary variables that minimizes

an objective function with polynomial factors up to quadratic terms. Formally

$$E(x) = \sum_i h_i x_i + \sum_{i < j} J_{ij} x_i x_j, \quad (2.2)$$

where  $x = x_i$  for  $i = 1, 2, \dots, n$  is the vector of binary decision variables,  $h_i$  is the linear term of  $x_i$ , and  $J_{ij}$  denotes the quadratic term between variables  $x_i$  and  $x_j$ .

The basic unit of Quantum information is the qubit, analogous to the bit in classical computing. Unlike bits, which can only represent a definitive state of 0 or 1, qubits can be in a superposition of states, representing multiple values simultaneously due to the principles of Quantum mechanics. This unique characteristic allows QA to tackle complex optimization problems with an approach that goes beyond classical methods. Qubits must be connected to create entanglement. This is achieved through couplers, which are essentially superconducting loops. When QA is programmed to solve a QUBO, the *bias* of a qubit  $q_i$  is set to  $h_i$ , and the *strength* of the coupler between  $q_i$  and  $q_j$  is set to  $J_{ij}$ .

In the context of QA, qubits are used to encode potential solutions to optimization problems. In the annealing process, the qubits are initially placed in the superposition state, with an equal probability of being either in the 0 or 1 state. While the annealing process evolves, the Quantum phenomena of tunneling and entanglement take place in this low-temperature environment, with the system converging slowly to a minimum of the QUBO model, which can be either a local or a global one. At the end of the anneal, each qubit has a classical state of 0 or 1 and is the solution to the QUBO problem.

When moving from a simulated annealing environment to a real-world Quantum processor, after the QUBO is formulated, it must be correctly programmed on the QPU before initiating the annealing process. To achieve this, each logical variable is mapped to a qubit, while the interactions between variables are mapped to couplers, the circuits that interconnect the qubits. The QUBO linear and quadratic terms set the qubit biases and coupler strengths, respectively. This process is called *embedding*.

In an ideal, fully connected QPU, each logical variable could be mapped to any qubit. QPUs are continually improving in terms of qubit connectivity; for instance, the qubit out-degree grew from 6 in Chimera to 15 in Pegasus and 20 in Zephyr [47]. Yet, the resulting QPU graphs are still sparsely connected due to the inherently technical complexity of providing complex qubit fabrics. Consequently, having only one-to-one mapping between a variable in the QUBO formulation and a qubit is impossible, and *chains* must be created.

Chains are sets of qubits that represent a single logical variable, ensuring the proper representation of the interactions in the initial QUBO. That is, neighboring variables in the QUBO formulation must also be so in the QPU, either directly or through a chain. When the QA process terminates, the state of the logical variable is determined by the majority vote of its chain's qubits' classical states. Intuitively, longer chains introduce

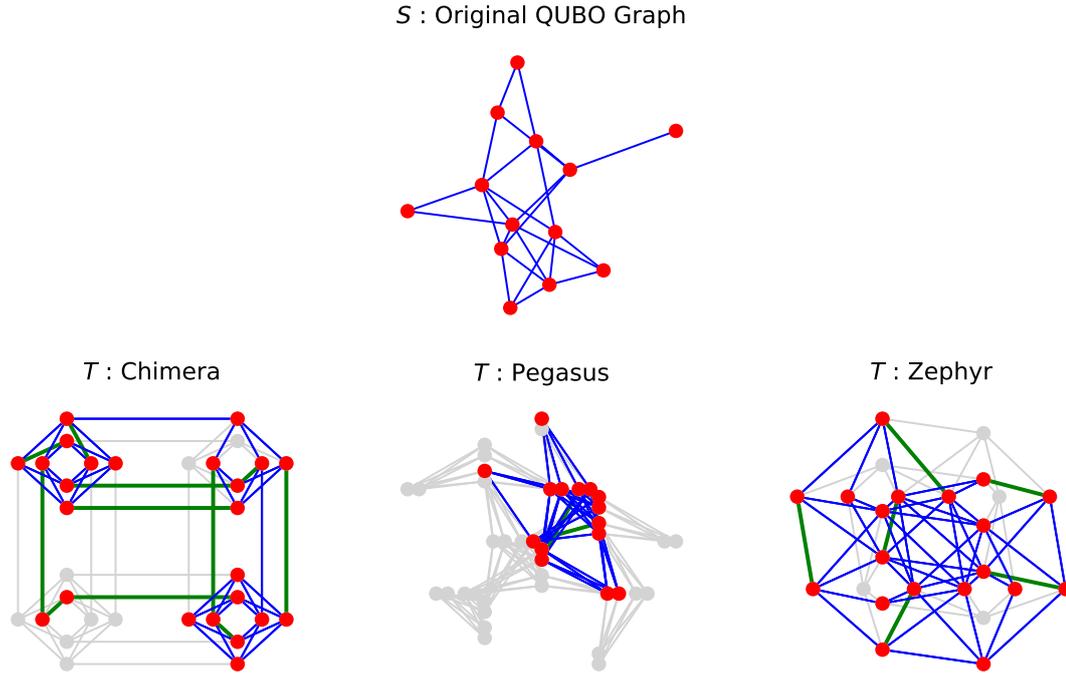


Figure 2.4: Embedding visualization of source graph  $S$  to Chimera, Pegasus and Zephyr target graph  $T$ . Red, blue, and green show the logical variables in  $S$  and occupied qubits in  $T$ , the edges of the source QUBO and the edges of the formatted chains, respectively.

larger sources of error in the annealing process, as more qubits and couplers (along with their inherent noise) must be used to solve the problem. In Fig. 2.4, I illustrate the embedding for the three different D-Wave target QPU architectures; in red, I denote the variables in the source graph  $S$  and the qubits in the target graph  $T$ , in green, I denote the couplers used for formatting chains, and in blue, the couplers as imposed by the original QUBO quadratic interactions. The top row depicts the original QUBO relations, while the bottom row depicts the embedding of the source QUBO onto the target graph.

The process of formatting chains to map a source graph  $S$  (*i.e.*, a QUBO graph in this case) to a target graph  $T$  (*i.e.*, the QPU graph) can be formulated as a *minor extraction problem*. In graph theory,  $S$  is a minor of  $T$  if  $S$  can be obtained by deleting edges and vertices or by contracting edges in  $T$ . Minor extraction is, in general, an NP-hard problem [48]. MinorMiner (MM) [49–51] is an iterative heuristic algorithm that solves the dual problem, determining which set of vertices (chain) in  $T$  corresponds to each vertex in  $S$  by taking into account the chain length of each source vertex. Once this mapping is obtained, the QUBO can be executed on the QPU, as the qubits and couplers can be programmed appropriately. In Fig. 2.4, I illustrate the embedding process with the green lines representing the chain links.

### 2.2.2 Quantum RAN Power Analysis

Performing wireless tasks such as FEC decoding in a Quantum computer has the potential to substantially impact the overall energy consumption footprint of the mobile network when compared with the traditional conventional-based computing systems. Google’s Sycamore Quantum computer reportedly has a power consumption of 15 kW [18], primarily attributed to its refrigeration and cooling unit (10 kW) and classical electronics (5 kW). D-Wave’s Advantage system reports a maximum power consumption of 25 kW, with cooling accounting for 15 kW [19].

Consequently, cooling is the primary source of the power consumption in Quantum systems [52], yet it is not expected to scale with the number of contained qubits [18, 52]. In contrast, conventional 5G base stations consume between 35 and 250 kW [17], depending on factors such as MIMO degree, number of receive antennas, and transmission bandwidth. These figures are significantly higher than those expected for Quantum systems, indicating the potential for substantial energy and Operating Expense (OpEx) reductions of up to 1,500% that can be attained by deploying Quantum processors for RAN.

## 2.3 Related Work

This section reviews key contributions from the literature on experimental wireless research platforms and strategies for optimizing the baseband signal processing pipeline. The focus is on approaches leveraging AI/ML for scheduling, CPU-aware design in vRANs, mobile networking software, and quantum baseband processing. Each subsection highlights limitations in the literature and explicitly maps them to the contributions of this thesis.

### 2.3.1 Mobile Networking Software

The rapid, easy, and cost-efficient deployment of end-to-end mobile networks using open source software has been studied in various works, and different implementations have been published mostly for large deployments. Works such as [53] and [54] leverage on the Open Air Interface (OAI) [55] as the RAN solution, using a hardware-based radio interface (i.e., USRPs) as the front-end, thus preventing the evaluation of more complex channel conditions and mobility scenarios (note that channel emulation is available with the latest version of OAI). Additionally, the core network is provided and orchestrated as a single container without management capabilities at NF granularity and without providing control and user plane separation. In [56], certain extensions are proposed in Kubernetes resources’ specification to support NFV loads.

In general, the experimental work in this field focuses on the RAN component.

For example, large-scale radio testbeds like Colosseum [57] have less focus on the core part, which is a fundamental aspect when testing end-to-end solutions. In Chapter 3, I demonstrate the design and management of an end-to-end cloud-native mobile network using open-source software that I used as a testbed for chapters 4 and 5. At the time this work was carried out, there had been no open source effort for cloud-native end-to-end mobile network stack emulation, exclusively conventional hosts, and no RF hardware front-end, which would be ideal for small experimental deployments for research purposes.

### 2.3.2 AI/ML MAC Scheduling

ML, particularly Deep Reinforcement Learning (DRL), has emerged as a promising method to address the complex, dynamic nature of MAC scheduling. These methods are especially appealing due to their ability to learn efficient policies in fast-changing wireless environments.

In [58], an actor-critic reinforcement learning agent is proposed to dynamically select the most suitable scheduling algorithm among several candidates, aiming to maximize overall QoS. The state representation includes factors such as the number of active users, packet arrival rates, Channel Quality Indicators (CQIs), and user-specific QoS metrics. The reward function quantifies the impact of scheduling decisions on meeting user QoS objectives.

Naparstek and Cohen [59] present a lightweight DRL algorithm tailored to the multi-user spectrum access problem. Each user independently selects a transmission channel at each time slot and receives an acknowledgment if the transmission succeeds. The state is defined by the history of selected channels and received observations, while the reward reflects the achievable throughput.

In [60], DRL is used to coordinate time-sharing between WiFi and LTE users. The proposed agent learns to select a time-splitting point based on past channel feedback, with the goal of allocating sufficient time to LTE users without violating WiFi throughput constraints. Although these works aim to enhance throughput and fairness, they generally overlook real-time processing deadlines and the challenges of operating within shared, resource-constrained vRAN infrastructures. On the contrary, ATHENA, as described in Chapters 4 and 5, is inherently built to consider the real-time nature of the vRAN system according to the instantaneous available CPU resources.

### 2.3.3 CPU-Aware RAN

On the CPU-aware scheduling front, Bhaumik et al. [26] investigated the computational resource demands of vRAN functions using a real-world testbed. However, their study did not account for the influence of user context or per-user scheduling decisions on computational load.

Subsequent works, such as [61] and [62], proposed analytical models to estimate computational load based on factors like SNR and MCS. Nevertheless, these models were not validated experimentally and may require adaptation to different hardware platforms. This limitation highlights the need for model-free approaches that can be directly deployed and validated on practical systems.

The architecture proposed in [63], **vrain**, tackles this challenge by jointly controlling CPU quota and MCS to optimize QoS, particularly by minimizing decoding error probability and buffer occupancy under resource constraints. Their contextual bandit formulation uses channel statistics and internal vRAN feedback (*e.g.*, from the LDPC decoder) to select MCS and PRB allocations. While evaluated on a real testbed, the system operates at coarse time granularity (seconds), which fails to capture rapid context changes in highly dynamic scenarios such as high-mobility users.

**BigStation** [29] introduced a distributed architecture for multi-user MIMO uplink processing, spanning FFT to decoding across multiple low-cost servers. The system reduced latency through a pipelined design, but was limited to small-scale MIMO scenarios (*e.g.*, 4G), due to resource constraints.

**Agora** [24] demonstrated a software-based approach for massive MIMO baseband processing on a single multi-core server, forgoing external hardware acceleration. Unlike **BigStation**, **Agora** employed data parallelism within a single frame rather than across frames, aligning well with modern high-core-count servers. However, this approach assumes access to high-end infrastructure and does not support resource sharing with third-party applications. **Savannah** [64] builds on top of **Agora** by modifying the SIMD operations in the channel precoder estimation and equalization stages, and by integrating ACC100, an ASIC approach for LDPC decoding. As an extension of **Agora**, it inherits the same limitations.

**Hydra** [65] builds on these efforts by minimizing inter- and intra-server communication overhead. It exploits channel similarity across adjacent subcarriers to reduce computational duplication. However, **Hydra** does not integrate MAC-layer scheduling into its architecture, nor does it consider the variable computational cost that arises from dynamic scheduling decisions. Furthermore, its design is tailored for distributed deployments, contrasting with the goals of infrastructure consolidation and cost sharing. Besides, all four **BigStation**, **Agora**, **Savannah**, and **Hydra** are not 3GPP-compliant but rather rely on a custom implementation of the physical low-level interface.

**Nuberu** [25], in contrast to previous work, builds upon open-source software and is validated on a real-world testbed. The authors redesign the uplink processing pipeline and associated MAC retransmission procedures to ensure sufficient headroom before the deadline hits. Additionally, they modify the MAC-layer retransmission mechanism to enable early discarding of potentially faulty transport blocks, thereby improving efficiency. However, their solution requires a complete reimplementaion of

the physical layer processing pipeline and is tightly coupled with a specific software stack, namely srsRAN [66]. This strong dependency on a particular implementation limits its generalizability and may hinder adoption by network operators and telecommunications vendors who rely on diverse, proprietary solutions. Finally, **CloudRIC** [8] introduces an RL-based MAC scheduler that distributes LDPC decoding tasks between CPU and GPU resources by controlling the number of PRBs requested by users. However, the authors focus solely on the interaction between the MAC layer and the LDPC decoder, employing a custom physical layer that accepts LLRs as input to the decoder. **CloudRIC** is not fully integrated with a 3GPP-compliant RAN stack, such as srsRAN, and therefore overlooks potential challenges associated with such integration. **ATHENA**, on the other hand, is fully integrated with the 3GPP-compliant srsRAN stack, offering a deployment option that is closer to real-world scenarios.

### 2.3.4 Quantum Baseband Processing

Seminal studies have already started exploring applications of Quantum computing to baseband processing. Proposals like **IoT-ResQ** [67], **X-ResQ** [68], and **QuAMax** [69] tackle the problem of MIMO detection using QA, whereas **HyPD** [70] considers Quantum for polar code decoding.

As far as the decoding of actual LDPC codes is concerned, **QBP** [16] was, to date, the first and only solution in the literature. While **QBP** matches the LDPC code design to the existing Quantum architecture to favor the implementability of the solution, the derived code significantly underperforms existing codes from the literature, such as Gallager [30], yielding an overall lower wireless performance. Additionally, **QBP**'s problem formulation, which also includes prior hyperparameter tuning depending on SNR, lacks formal guarantees that the minimum energy solution is the maximum likelihood solution of the fundamental decoding problem. **QBP** is the main baseline of the work in Chapter 6, where further insights will be given.

Further variants of **QBP** have focused on complementary aspects. In [71], the authors post-process the Quantum computer results by discarding invalid codewords and identifying the one closest to the received vector. The study in [72] evaluates different post-processing techniques, including the minimum energy solution or the solution with the highest frequency. In contrast, the authors in [73] consider the effects of the Rayleigh channel, unlike the Additive White Gaussian Noise (AWGN) channel, which was used in previous works. However, all works in [71–73] do not address the fundamental limitations in **QBP** and share the same weaknesses highlighted above, that is, they employ an underperforming problem formulation for FEC decoding and utilize the same code design algorithm. In Chapter 6, I detail significant improvements that I achieved in that regard, demonstrating superior performance to **QBP**.

Table 2.1 summarizes the aforementioned related works, highlighting their conceptual

limitations or gaps, and maps them to the three main contributions of this thesis.

<b>Related Work</b>	<b>Limitations / Gaps</b>	<b>Thesis Contributions</b>
[53, 54, 56, 57]	No small-scale, end-to-end cloud-native network stack	End-to-end cloud-native mobile network testbed using software RF front-ends running on commercial laptops/hardware (Chapter 3)
[8, 24–26, 29, 58–65]	Real-time deadlines and dynamic vRAN CPU resource allocation are overlooked. Coarse granularity, lack of experimental validation with a 3GPP-compliant software stack	CPU-aware, deadline-conscious MAC scheduling validated on end-to-end cloud-native 3GPP-compliant testbed (Chapters 4, 5)
[16, 67–73]	Underperforming quantum LDPC decoders, formulation not providing the optimal solution	Quantum-compatible LDPC decoding framework with improved performance (Chapter 6)

Table 2.1: Mapping of related work and its limitations to the thesis contributions.

# 3

## Orchestration of Cloud-Native Mobile Networks

---

### Disclaimer

This chapter is based on the following publication:

- **Nikolaos Apostolakis**, Marco Gramaglia and Pablo Serrano, "Design and Validation of an Open Source Cloud Native Mobile Network," in *IEEE Communications Magazine*, vol. 60, no. 11, pp. 66-72, November 2022, doi: 10.1109/MCOM.003.2200195

Minor changes in structure and notation have been made to improve readability and consistency within this thesis.

The softwarization of mobile NFs allows network operators to reduce their operational expenditure by using open software and also enables researchers and practitioners to experiment with new network functionalities. This software-driven transition is being accelerated with the introduction of cloud-native architectures [74], which greatly increases the flexibility and scalability of the software. In what follows, I detail the different components of the cloud-native mobile network design, several of them chosen among the set of open-source solutions that have become available during the last few years. I review every component of the cloud-native RAN and core design, which will serve as a reference architecture for Chapter 4.

### 3.1 Cloud-Native Core and RAN Design

The different components of the design are illustrated in Fig. 3.1. For orchestration, Kubernetes is the most prominent, open-source system for deploying and managing containerized applications. For the core network implementation, Open5Gs is selected due to its adaptability to the cloud native paradigm. For the UE and the RAN, srsRAN provides a full mobile network stack implementation, completely open-source. Furthermore, the srsRAN community is very active, and its contributors keep intensively

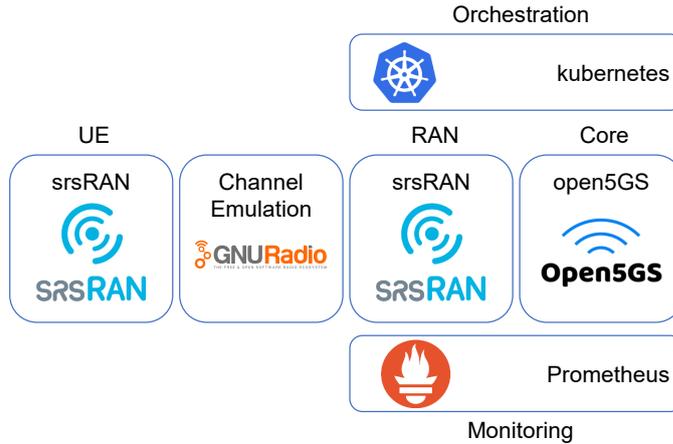


Figure 3.1: The solutions that build the end-to-end cloud-native architecture.

improving the product to keep up with the latest 3GPP standards. To simplify the deployment and support repeatability, I developed a simple channel emulator, which enables deploying a fully programmable end-to-end network without requiring expensive hardware. Finally, I chose Prometheus, one of the monitoring solutions promoted by the Cloud Native Computing Foundation (CNCF) [75], as the monitoring framework to collect the status of the whole system. In what follows, I detail these components.

### 3.1.1 Management and Orchestration

Kubernetes (also known as `k8s`) is a container orchestration platform, widely adopted in the cloud computing domain for more than five years. A typical cloud native application consists of a set of container images (*e.g.*, Docker images) that are deployed and managed by Kubernetes, using configuration files that describe the functionality and the services they provide. The minimum deployment unit on Kubernetes is the *pod*, which may contain more than one container, while the intercommunication between pods and the outside world is achieved using different types of *services*. Its scaling and redundancy capabilities have already been demonstrated in production environments, and it is envisioned that telco operators will rely on Kubernetes for next-generation networks, motivated by the need to make more efficient use of the heterogeneous virtualized infrastructure (which spans over central and edge data centers). To manage the additional management complexity, a huge ecosystem of open source projects that extend the possible operational tasks has been created under the umbrella of the CNCF.

### 3.1.2 Core Network

Open5Gs is a very popular open-source implementation of a mobile network core. Written in C, it stands as a reference among researchers and mobile telecommunications

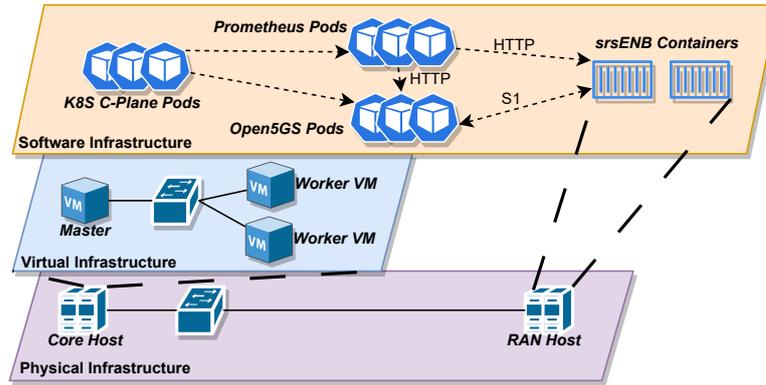


Figure 3.2: Core and RAN Infrastructure setup. The physical layer hosts the infrastructure supporting both the core and RAN functionalities. The virtual layer runs virtual machines for the control- and user-plane network functions. The software layer includes the Kubernetes control-plane Pods (*K8S C-Plane Pods*), Prometheus Pods, Open5GS Pods (encompassing both 5GC control- and user-plane components), and the srsENB Docker containers.

practitioners for experimentation and future enhancements. It contains the most important components of the 5GC and 4G Evolved Packet Core (EPC) with Control-User Plane Separation (CUPS) [76], meaning it can operate on both 5G Non-Standalone (NSA) and Standalone (SA) modes, as it can serve both 4G E-UTRAN NodeB (eNB) and 5G gNB. Its straightforward build procedure makes its deployment in small-scale private networks very easy, while its modular architecture could be considered a natural candidate for running it in microservices-based cloud-native environments powered by Kubernetes. As illustrated in Fig. 3.2, Open5Gs control-plane and user-plane pods will be deployed over the virtual infrastructure.

### 3.1.3 RAN

srsRAN [66], formerly srsLTE, is another open-source software framework, developed by Software Radio Systems, that provides the functionality from the PHY up to RRC layer for eNBs/gNBs, while also supporting 4G/5G UEs. It is written in C/C++, and its configuration parameters cover a wide range of base station and UE possible configurations. Regarding the RF interface, its contributors have developed drivers for various commercial hardware RF frontends like USRP, Soapy SDR, and BladeRF.

Additionally, srsRAN provides a software RF-frontend based on ZeroMQ, an open-source message queueing library written in C. When using this driver, the transmitted time (post IFFT) samples between the UE and base station are transferred over various transport methods, like inter-process communication or TCP sockets. Choosing this driver avoids the need for high expertise in RF channel configuration and facilitates the introduction of researchers who want to simulate a radio access network environment,

but whose RF channel is not their main area of interest, or would be reluctant to invest in actual hardware transceivers. As shown in Fig. 3.2, srsRAN was executed as Docker containers on top of the bare-metal physical infrastructure. The eNB version of srsRAN was selected to perform the handover experiments, described in Sec. 3.3, which were not supported by srsRAN for the gNB. Next, I discuss how to take advantage of ZeroMQ to implement a channel emulation model.

### 3.1.4 Channel Emulation

The use of ZeroMQ enables running the network in a fully softwareized way. Furthermore, it enables the emulation of complex topologies via programming (*e.g.*, arbitrary complex topologies can be created by dynamically connecting endpoints, as they do in large-scale emulators using hardware in the loop [57]). The main challenge when emulating complex mobility scenarios relies on the handling of time samples directly, *e.g.*, sending low power symbols from the eNB to trigger a Handover Request from the UE.

Following the srsRAN Documentation, GNU Radio Companion (GRC) was used, which is another open source project for Software-defined Radio (SDR), and, among others, contains modules for the ZeroMQ library. I coded in Python a `GRC Broker`, located between the UE and the cells of the eNB(s) implementing the RF interface, as illustrated in Fig. 3.3. This module intercepts the transmitted time samples and performs operations on them to emulate the channel condition for the two traffic directions:

- In the downlink direction, to simulate the distance between the UE and the cells, I made use of multiplier blocks followed by an adder. Multiplier blocks multiply the time-domain cells' transmitted samples with a constant gain value in the range  $[0, 1]$ , adjusting UE's perception of the cells' signal strength. The adder block simply performs the superposition of the modified samples, as would be the case in the real environment.
- In the uplink direction, I used a throttle block that re-transmits the I/Q symbols towards the cells of the eNB(s).

Leveraging this setup, I evaluated different mobility and load scenarios (discussed in Section 3.3). Intercepting the time samples directly has been instrumental in performing different SNR measurements in Chapter 4.

### 3.1.5 Monitoring

In order to have a consistent view of the mobile network's current status in a unified platform, I deployed the cloud-native version of Prometheus (*kube-prometheus-stack*) in

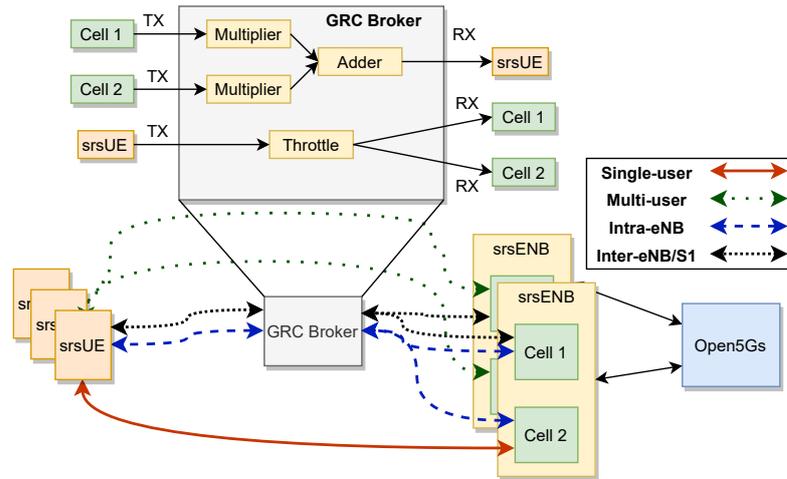


Figure 3.3: The evaluated mobility and load scenarios between UEs and eNBs. The UEs connect with the eNBs through the GRC Broker, that enables intra- or inter-eNB handovers. The eNBs/gNBs connect to the Open5Gs.

the same Kubernetes cluster used for the Open5Gs. In this section, I describe the steps taken to monitor both Core and RAN through Prometheus.

### 3.1.5.1 Core Network

When *kube-prometheus-stack* is deployed in a Kubernetes cluster, by default it installs exporters in all the Kubernetes's controller and worker nodes, which collect and expose metrics like CPU usage, RAM, Networking status, and IOPS on different granularity levels (node, pod, container). By exporting those metrics, the health of the virtual infrastructure running the different core NFs is monitored, as depicted in Fig. 3.2. In Fig. 3.2, the Kubernetes control-plane components are shown as *K8S C-Plane Pods*, while the Open5GS control- and user-plane components are collectively labeled as *Open5GS Pods*. Prometheus operator pods have set those exporters as targets and will start polling them in fixed intervals using HTTP. Those metrics are directly accessible either through the Prometheus WebUI or Grafana, a visualization tool, using PromQL, a functional query language.

### 3.1.5.2 RAN

srsRAN produces an array of metrics for the lower layers of the networking stack, which can be exported to the console or a text file. However, srsRAN does not have an agent acting as a metrics exporter toward 3rd party software. Therefore, I modified the source code of the srsENB in order to develop and integrate a new one into the final binary<sup>1</sup>. The first step towards this direction was to integrate *lighttpd* project, a lightweight open-

<sup>1</sup>The code is available on <https://github.com/kaposnick/srsLTE>

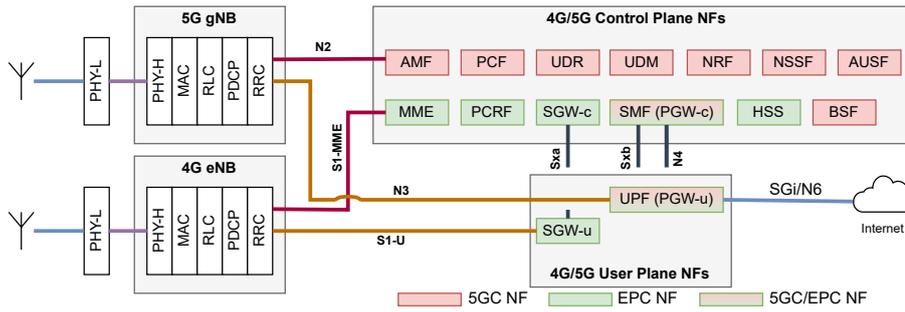


Figure 3.4: The network functions included in Open5Gs and srsRAN.

source HTTP server written in the C programming language. Then, I developed the REST API through which the metrics, in key-value pair format, will be exported to the Prometheus operator. Afterwards, I developed Kubernetes manifests which comprise a pod, whose role is to proxy the HTTP requests received by the Prometheus operator, towards the end srsENB process. Those manifests are deployed to the Kubernetes cluster with the IP address of the srsENB as an input parameter. Last, facilitating the discovery capabilities of Prometheus, a *ServiceMonitor* configuration resource was deployed, which automatically discovers the 'proxy' pods existing in the cluster and sets them as Prometheus targets.

Thus, for every srsENB process instantiated, an auto-discovery service is performed and, once it is completed a few seconds later, the eNB starts getting polled (or scraped, using the Prometheus terminology).

## 3.2 Network Deployment

One of the main advantages of cloud-native solutions is their extreme flexibility, thanks to the easy templating and parametrization of the software components. I used Kubernetes manifests and Helm charts<sup>2</sup> to perform the mobile network configuration.

### 3.2.1 Kubernetes Manifests

To be able to manage very heterogeneous software components, Kubernetes needs to abstract the specific parameters associated with them. To this aim, the so-called *Manifests* are used to define the information model (*i.e.*, the parameters associated with each function) for each NF. Thus, I developed the Kubernetes resource files for each software component in the network architecture. Open5Gs provides software implementing the NFs for both EPC and 5GC under the same umbrella, as depicted in Fig. 3.4:

- *Control-plane functions*: MME, HSS, PCRF, SGW-C, PGW-C (for the EPC); AMF, SMF, UDM, UDR, NRF, NSSF, AUSF, PCF, BSF (for the 5GC).

<sup>2</sup>Kubernetes manifests and Helm charts are available on <https://github.com/kaposnick/k8s-open5gs>

- *User-plane functions*: SGW-U, PGW-U (for the EPC), and UPF (for the 5GC).

Hence, manifests include information such as the exposed ports, the internal interconnections, and the software components to be deployed by Kubernetes. Each NF is instantiated within the context of a *StatefulSet* controller, instead of a standalone *Pod*, so that it is appropriately managed in case of termination. Additionally, to support realistic use-case scenarios, where control plane and user plane NFs should be closely located but in distinct nodes, I set the *nodeSelector* attribute to `mobile-core:control` for control plane NFs and `mobile-core:user` for user plane NFs. This attribute is enforced by Kubernetes at the scheduling stage.

For the inter-pod communication, I used the *ClusterIP* service type, so that the corresponding service pods are accessible only within the Kubernetes cluster (preventing external access), while for the interfaces exposed towards the RAN (*i.e.*, S1-MME/N2 by the MME/AMF, S1-U/N3 by the SGW-U/UPF) I used the *LoadBalancer* service type. This type of service is used in cloud environments to enable the external connectivity towards the cluster's pods, and typically uses a load balancer at the frontier of the cloud DC.

### 3.2.2 Helm Charts

The deployment stage of a cloud computing service (a mobile network core, in this case) has proven to be prone enough to human errors for several reasons:

- A considerable amount of Kubernetes manifests ( $\sim 3$  manifests per NF) has to be applied for every single iteration of an experiment. During the deprovisioning of the service, the reverse procedure has to be executed, taking care to de-instantiate all the NFs.
- A few manifest attributes have to be provided at the initial stage, as they depend on the current state of the environment. Such attributes are the IP addresses that must be accessible by the RAN. Statically defining them, such as in a production environment like an operator's Data Center (DC), is a solution that limits the flexibility of the deployment in a local environment, as they depend on the Network Interface Card (NIC) IPs, which are obtained through DHCP and are renewed upon reboot. Those IPs are used for setting the *LoadBalancer* service configuration as described above and the IP address that the UPF advertises to the SMF (and consequently to the AMF and the gNB) when addressing a service request.

Thus, as already done by major cloud computing services, Helm charts can automate those configuration tasks by storing parameterized Kubernetes manifests along with a

single configuration file (containing all the global parameters) in a centralized location (this chart can be uploaded to an online registry for accessibility reasons, as for the case of the Docker images). Then Helm automatically generates the equivalent Kubernetes manifests and afterward provides them to the Kubernetes API server. This results in a single command executed for (de)provisioning of the entire core network, thus greatly improving the automation.

### 3.3 Experiments

The Kubernetes cluster consists of a KVM cloud platform using Ubuntu 18.04.6. Ubuntu 20.04.1 LTS is the image of all the nodes of the cluster (one master and two workers, as illustrated in Fig. 3.2), with Docker 20.10.8 as the container runtime, as well as `kubelet` and `kubect1`. The control plane was set using the `kubeadm` utility and Flannel as the Container Networking Interface (CNI). The NFs were connected to an internal network for control plane and inter-pod communication, and NAT forwarding is enabled for management purposes through SSH. The two worker nodes had an additional network adapter of type `Bridged Adapter` attached to another Ethernet Adapter of the host machine, exposing the N2 and N3 interfaces towards the gNB(s). The two worker nodes were also annotated with the labels `mobile-core:control` and `mobile-core:user`, which are taken into account during the deployment stage of the NFs. The RAN (`srsENB/srsUE`) is set up using Docker in a commercial laptop with the 4 cores CPUs at 1.8 GHz and 16 GB of RAM. Each `srsUE` process runs in a separate networking namespace in order to have distinct routing tables.

Below there two different scenarios to validate the adequate behavior of the developed modules. For each considered scenario, the Open5Gs helm chart had to be deployed from scratch to fully decommission software components from previous runs. Also, the UDR database had to be populated with UE information for authentication purposes. At each chart deployment, the core network reached a healthy state after about 2 minutes, with several pods restarted due to unfulfilled interdependencies: for instance, the UDM container has to wait for the UDR database container to start up; otherwise, it is restarted. To check the end-to-end state of the core network, I access the Prometheus REST API (which exposes the retrieved metrics, as discussed before). I then analyze the performance of the system for the following scenarios. I remark that, in addition to these two scenarios, the platform can be used for several other research activities, *e.g.*, prototyping new radio schedulers (as in Chapter 4), developing novel fine-grained orchestration algorithms involving CPU pinning or thread-level quota, or building on the GRC broker for the emulation of more complex scenarios variable.

### 3.3.1 Dynamic NF orchestration

In this case, the objective is to stress the u-plane NFs, without disrupting UE's throughput. I first deployed one mobile network instance, consisting of one 5GC, and one gNB with one associated UE, and configured the UE to run an `iperf` test towards an `iperf` server connected to the UPF through the N6 interface. In Fig. 3.5, the amount of CPU resources consumed by the different components via the Prometheus exporter is measured. According to the results, the user plane function occupies most of the CPU, while the rest of the NFs have an almost negligible footprint.

To illustrate the dynamic 5GC NF orchestration capabilities of the system, which may be leveraged by *e.g.*, AI algorithms (such as the one presented in Chapter 4 but in the RAN part), in combination with the monitoring capabilities, I trigger the instantiation of another network at approx. 1000 s, with a UE connected to the second 5G instance. In real-world 5G deployments, dynamic instantiation of an additional 5GC instance becomes necessary in scenarios involving rapid changes in network load, geographic distribution of users, or service isolation requirements. By triggering the launch of a second 5GC instance, I emulate such scenarios, validating the orchestration platform's ability to scale core components responsively.

While the instantiation of the second 5GC instance was triggered manually in this scenario, it effectively simulates the dynamic behavior expected from an AI-driven or policy-based orchestrator. The system architecture includes real-time monitoring via Prometheus and data shippers, which could be leveraged to enable fully autonomous decision-making based on NF load or performance metrics. Thus, this experiment demonstrates the platform's readiness for dynamic orchestration, even if the control loop was not closed in this instance.

In this context, the performance level refers to maintaining stable throughput for all UEs and avoiding performance degradation (*e.g.*, throughput drops) as new network instances are instantiated. During the experiment, I observed that the first UE's throughput remained unchanged, and the second UE maintained a stable data rate for its entire session, indicating that service quality was preserved across both core instances. According to these results, I demonstrate that this platform can be used to emulate *e.g.*, horizontal scaling scenarios, where additional resources are added to guarantee a given performance level.

As Fig. 3.5 illustrates, the instantiation of the second network causes a modest increase in total CPU usage, peaking at around 18%. This value reflects the cumulative overhead of both 5GC instances, where, as expected, the majority of CPU resources are consumed by the user-plane UPF components. The remaining NFs, such as the AMF, SMF, and others, exhibit minimal resource usage in comparison. This observation aligns with the expected behavior of user-plane-heavy workloads, and underscores the importance of UPF-aware orchestration policies in real deployments.

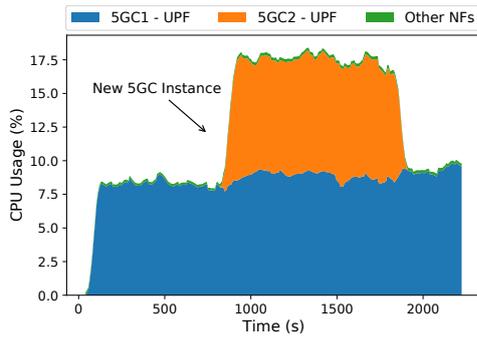


Figure 3.5: CPU Utilization of two 5G core instances.

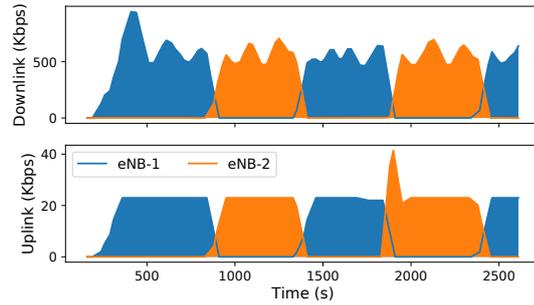


Figure 3.6: RAN throughput.

### 3.3.2 UE Mobility

In this section, I illustrate the ability of the deployment to emulate situations where the UE performs handovers between different cells. By intercepting the transmitted time samples, UE’s perceived SNR at the two base stations changes, which triggers the handover mechanism. For this experiment, I relied on 4G connectivity, since the srsRAN at the time these experiments were performed (*i.e.*, 22.04) did not support handovers on the 5G network.

I deployed a mobile network instance, with one EPC and two eNBs, where each eNB generates two cells. I then deploy the UE, which is associated with the first cell. Like in the previous case, each UE generates bi-directional `iperf` traffic towards a server, but now connected to the PGW-U.

From the above scenario, I trigger the handover by taking advantage of the GRC broker to emulate a change in the channel conditions, effectively changing the UE’s SNR. More specifically, I program the GRC broker to emulate the movement of UE between the two cells by judiciously changing the received power from the UE to each eNB (in the uplink) and from the eNBs to the UE (in the downlink). This is done by modifying the gains that multiply the samples from the eNBs, and throttling the samples from the UE (see Fig. 3.3), achieving the effect of slowly moving away from one eNB and getting closer to the other one. The inverse procedure is applied to emulate the movement back to the previous cell, and the whole process is repeated indefinitely.

Fig. 3.6 depicts the resulting downlink (top) and uplink (bottom) throughput at the MAC layer of the two eNBs. Firstly, it is worth noting the relatively small throughput values, which are caused by the emulation of the channel that “enlarges” the duration of the slots (thus decreasing the actual throughput). Secondly, the figure clearly illustrates how the throughput moves between eNBs, where approximately only one eNB is sending and receiving traffic at a time. While temporary drops in throughput can be observed during handovers, the throughput returns to its previous level immediately

after the handover completes. This behavior demonstrates that the testbed is capable of reliably emulating handover scenarios. The significance of these transient throughput drops depends on the specific service requirements. For example, in latency-sensitive applications, such drops might trigger TCP timeouts or affect QoS. However, quantifying the application-level impact of such fluctuations is beyond the scope of this work. Instead, the focus here is on validating the functional correctness and flexibility of the handover emulation, which serves as a solid foundation for developing and testing mobility-aware orchestration mechanisms.

These results showcase the flexibility of the implementation of the GRC broker, which can be used and extended to support more complex radio scenarios, and be used as a sandbox for the development of mobility-triggered mechanisms (*e.g.*, follow-the-load orchestration) in a controlled environment, as shown in Chapter 4.

## Summary

In this chapter, I presented the design and deployment of a fully cloud-native mobile network, encompassing both the 5G Core and the RAN, running entirely on COTS hardware and open-source software. The implementation combined Kubernetes for orchestration, Open5GS for the core, srsRAN for the RAN, and Prometheus for monitoring, while using a lightweight channel emulator to remove the dependency on proprietary RF front-ends.

The main conclusion is that cloud-native architectures can serve as a practical foundation for end-to-end mobile networks without requiring specialized hardware. To the best of my knowledge, at the time this work was carried out, this study represented the first open-source and reproducible implementation of such a system, which I have made publicly available. This contribution democratizes access to research on cloud-native mobile networking, allowing practitioners and researchers to prototype, experiment, and evaluate new ideas at low cost.

This testbed forms the basis for the following chapters 4 and 5, where I build on it to investigate CPU-aware scheduling mechanisms in vRAN environments.



# 4

## ATHENA: Resource-Aware MAC Scheduling

---

### Disclaimer

This chapter is based on the following publication:

- **Nikolaos Apostolakis**, Marco Gramaglia, Livia Elena Chatzieftheriou, Tejas Subramanya, Albert Banchs and Henning Sanneck, "ATHENA: Machine Learning and Reasoning for Radio Resources Scheduling in vRAN Systems," in *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 2, pp. 263-279, Feb. 2024, doi: 10.1109/JSAC.2023.3336155

Minor changes in structure and notation have been made to improve readability and consistency within this thesis.

Despite the decoupling of the underlying networking infrastructure from the software-based implementation in vRANs, their operation still presents some fundamental problems, such as the optimal interplay between the cloud infrastructure and the vRAN software. In this chapter, a deep learning model, namely **ATHENA**, for radio scheduling is contemplated. **ATHENA** aims to maximize the uplink throughput by jointly controlling the radio resources, considering the congestion incurred by third-party applications running on the shared compute infrastructure and the strict timelines imposed by 5G, which should be respected to maintain a standard quality of service.

In Sec. 4.1, I introduce the system model components and the vRAN radio resource control scheduling problem. In Sec. 4.2, I describe in detail **ATHENA** framework and the design choices. In Sec. 4.3, I elaborate on the implementation details, crucial for assessing the solution using the experimental testbed, while in Sec. 4.4, I evaluate **ATHENA** radio scheduler against the srsRAN's default one, demonstrating its advantages.

Table 4.1: Notation Table.

Short	Expansion	Variable	Description	Set
UE	User Equipment	$u$	User	$\mathcal{U}$
MCS	Modulation Coding Scheme	$m_u$	User's $u$ MCS	$\mathcal{M}$
PRB	Physical Resource Block	$n_u$	User's $u$ PRBs	$\mathcal{N}$
TBS	Transport Block Size	$l_u$	User's $u$ TBS	
CRC	Cyclic Redundancy Check	$r_u$	User's $u$ CRC	
		$\bar{c}_u$	User's $u$ Channel Conditions	
		$t$	Slot	
		$\beta$	Congestion Factor	$\mathcal{B}$

## 4.1 System model

This section introduces the system model that underpins the design and evaluation of ATHENA. The goal is to capture both the wireless communication aspects of UL transmissions and the computing constraints of vRAN deployments, so that the resulting framework can reason jointly about radio resource allocation and processing feasibility. Unlike traditional MAC schedulers, which only optimize for spectral efficiency or fairness in the wireless domain, ATHENA incorporates the decoding effort and platform congestion into the scheduling decisions.

To this end, I first outline the components of the model, whose notation is provided in Table 4.1, including the gNB, users, and resource allocation elements. I then formulate the scheduling problem that arises when computing and radio constraints interact in congested vRAN environments, where meeting strict uplink deadlines is particularly challenging. Finally, I emphasize the broader implications for uplink-intensive applications such as live video streaming and AR/VR, which put sustained pressure on the uplink pipeline and could benefit the most from ATHENA's design.

### 4.1.1 Model Components

The system consists of a set  $\mathcal{U}$  of UEs  $u$  and a gNB. At each slot  $t$  in Frequency Division Duplex (FDD), or at each UL slot in case of Time Division Duplex (TDD), the gNB allocates to each UE  $u$  some PRBs and a specific MCS for its UL data transmission. This work focuses on the UL transmission since, UL pipeline creates the major computing load for the gNB [8, 25, 26] and it is one of the most important factors to consider for vRAN systems.

At each slot, the MAC scheduler assigns grants to the selected UEs, where each grant consists of the number  $n_u$  of PRBs and MCS  $m_u$  for user  $u$ . The combination of  $(n_u, m_u)$  yields a TB of TBS  $l_u$ .

### 4.1.2 Problem: Resource Scheduling in Congested vRANs

In Chapter 2, the hard deadline for completion of UL processing tasks was introduced. Let this deadline be  $J - 1$  slots, then the UL processing tasks have to be completed within  $J$  slots after the reception at slot  $t$ . I now discuss the computing footprint of a vRAN system and why it is paramount in a vRAN system to also consider the computing effort of radio resource scheduling decisions.

#### 4.1.2.1 Software Implementations and Complexity

LDPC decoding is the most time-consuming block in the UL processing pipeline [25, 26]. The implementation of the procedure can be represented by two nested loops (Sec. 2.1.1.2): an inner one that iterates over the LLR values, maximizing their likelihood, and an outer one that loops until the target likelihood is reached (or a maximum number of iterations  $I_{MAX}$ ). Thus, the processing latency  $T_{dec}$  of the decoding task is affected mainly by: (i) The TBS  $l_u$  as it sets the number of iterations needed to process all the LLR (*i.e.*, a larger TBS implies more iterations in the inner loop), and (ii) the relation between the channel conditions and the selected MCS  $m_u$ , that I will next explain.

The actual number of iterations of the decoding algorithm depends on the selected  $m_u$  and the experienced SNR. With low  $m_u$  and high SNR, the decoding ends within a few iterations of the outer loop with a very high probability. Otherwise, more of them are needed up to the point that the selected  $m_u$  cannot be decoded under the given channel conditions, and hence even after  $I_{MAX}$  iterations, the TB cannot be decoded, and a NACK is sent to the UE to trigger its retransmission. This value is implementation-dependent. In the software implementation used throughout this chapter,  $I_{MAX} = 8$ .

Given the time-sharing nature of cloud systems, larger  $T_{dec}$  values imply a higher probability of suffering computing fluctuations (*e.g.*, due to context switching). In this case,  $T_{dec}$  becomes less deterministic and thus less suitable for the frame decoding tasks. Fig. 4.1 shows  $T_{dec}$  vs. TBS (obtained with the testbed discussed in Chapter 3). It illustrates the elapsed time for full decoding, under varying channel quality conditions, and different competing loads (modeled by the variable  $\beta$ , later introduced).

When no competing load is present (green line in Fig. 4.1), the variability of  $T_{dec}$  increases linearly with the TBS. In the medium and high congested scenarios, though, the trends of both average and variability of  $T_{dec}$  grow non-linearly. This effect becomes evident when the vRAN system is competing for computing resources with others, and only a reduced amount of frames can actually be decoded within the default deadline of  $J - 1$  slots. In 5G systems, this deadline is configurable but is typically set to 3 ms for mobile broadband traffic [8]. The same holds for 4G systems, though the 3 ms deadline is fixed, as the eNB must provide feedback to the UE precisely 4 ms after receiving the PUSCH [77], corresponding to four slots.

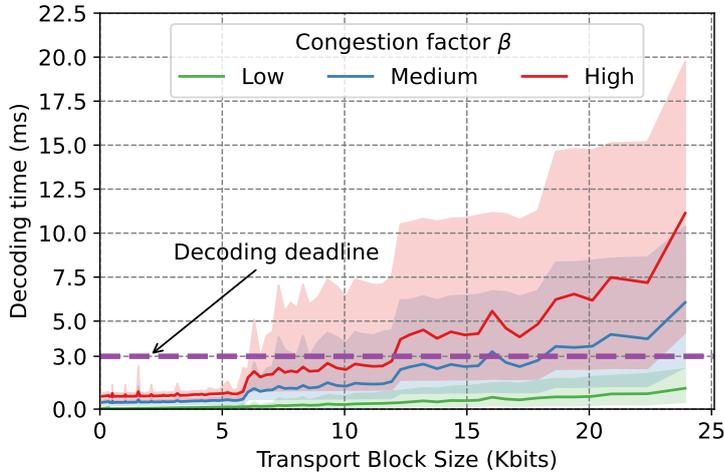


Figure 4.1: Decoding time  $T_{dec}$  for  $MCS \in [0, 24]$ , various eligible numbers of UL PRBs, under different SNR conditions, and three congestion factors  $\beta$ . Solid lines denote the averages, while shaded areas depict the 5th to 95th percentile.

#### 4.1.2.2 CPU Congestion

On cloud computing systems, such as those employed by vRANs, different execution threads share the same computing platform (*i.e.*, CPU cores) to perform tasks. The element that is in charge of multiplexing computing resources among the different processes is the operating system’s CPU scheduler, which assigns CPU quota to processes according to some periodicity rules (*e.g.*, fairly sharing the amount of CPU time across processes) or when the process would not efficiently use the resources because it has to wait for the completion of another operation (*i.e.*, upon an I/O operation or a memory cache miss).

To capture this aspect effectively, I denote *congestion factor* by  $\beta \in [0, 1]$ , mimicking the overhead incurred by other competing processes by slowing down the existing ones. This allows the decoding process and the number of competing processes to be described independently of the computing platform that is used, working with any CPU clock time.

$\beta = 0$  indicates that there are no competing processes and that vRAN operates at a full CPU speed, while  $\beta = 1$  indicates a very slow and crowded computing platform. More details about the implementation of  $\beta$  are provided in Sec. 4.3. The congestion factor  $\beta$  takes into account the dynamic nature of the data center that the O-RAN Cloud (O-Cloud) is running on. That is, this is not a priori fixed and can be tracked or measured (Sec. 4.3) or can be configured by preference by an orchestration algorithm (Sec. 5.2.3).

Having said that, the role of the MAC scheduler becomes crucial. CPU-blind MAC schedulers, typically deployed, allocate resources, affecting the actual data transmitted, that do not manage to be processed within the technology timelines. The MAC schedulers

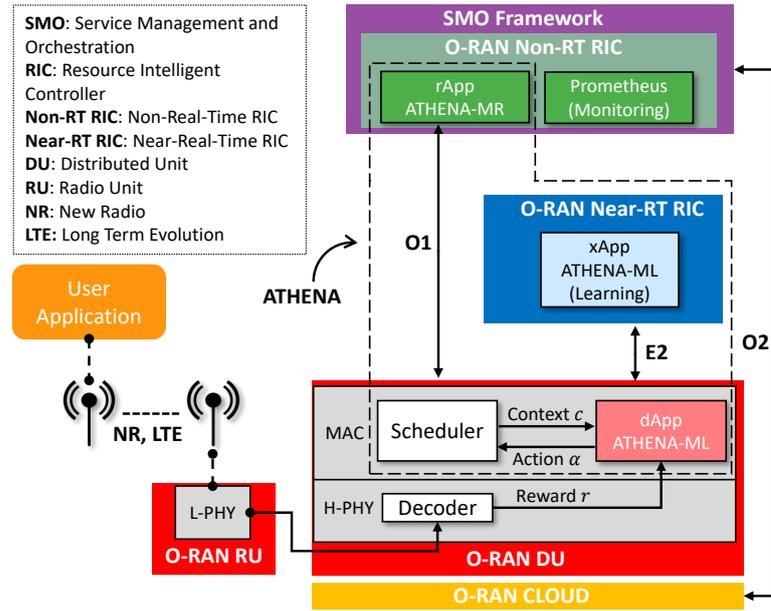


Figure 4.2: ATHENA and its integration with the most important modules of the 3GPP and O-RAN architectures. ATHENA framework comprises *i*) ATHENA-ML resource controller that integrates with a user selection component to provide a fully-fledged MAC scheduler, and *ii*) ATHENA-MR engine that (re-)orchestrates O-Cloud resources to optimize the O-DU behavior.

must become congestion-aware, controlling the granted radio resources and ensuring the reliability of the whole system.

### 4.1.3 Relevance for Uplink-Intensive Use Cases

In realistic network deployments, many emerging applications, such as live video streaming [78–80] and augmented/virtual reality [81], generate substantial traffic in the uplink direction. These applications require sustained allocation of UL PRBs at high modulation and coding schemes, thereby increasing both the TBS size and the computational burden at the gNB. In conventional CPU-blind schedulers, this often results in buffer build-ups, retransmissions [25, 80, 82], and violations of latency deadlines under high load [8, 25]. By contrast, ATHENA explicitly accounts for the decoding effort and congestion factor  $\beta$  when assigning uplink resources, effectively trading off throughput against the risk of decoding failures. This enables more predictable performance for uplink-heavy applications even under constrained or fluctuating cloud resources. In practice, this means that while ATHENA may allocate fewer resources to certain users compared to an aggressive throughput-oriented scheduler, the allocated grants are more likely to be successfully decoded within the deadline, and thus, not affect the delay of the upper-layer applications (such as TCP-oriented ones).

## 4.2 ATHENA

In this section, I contemplate on ATHENA framework, which, as depicted in Fig. 4.2, consists of two main functional blocks: *i*) ATHENA-ML resource controller that adjusts the radio resources of the scheduled users to adapt to the dynamic changes of the execution environment and integrates with the user selection component, and *ii*) ATHENA-MR that orchestrates the computational resources of the O-DU. Both the user scheduler and ATHENA-ML operate at a millisecond timescale [21], providing uplink scheduling grants to UEs. I consider that any user selection component could be integrated in ATHENA (*e.g.*, round-robin, proportional fair), and I study the radio resource controller component. This chapter focuses on ATHENA-ML, while Chapter 5 details ATHENA-MR.

ATHENA-ML, a Contextual Bandit (CB) learning algorithm, creates a closed control loop taking actions based on contextual information coming from the RAN, *e.g.*, the UE-associated wireless channel conditions, and from the cloud provider’s data center monitoring system, *e.g.*, the current  $\beta$ . ATHENA-ML then transforms this information into policies, optimizing RAN performance to the hard time deadlines imposed by the computing platform.

The problem introduced in 4.1 can be formulated as a CB problem, in which in each episode  $t \in T$ , the agent receives the current context  $c^{(t)}$  as a feature vector drawn from a context distribution  $\mathcal{C}$ . The agent then chooses and executes an action  $\alpha^{(t)} \in \mathcal{A}$ , where  $\mathcal{A}$  is the action space, and receives a reward  $r(c^{(t)}, \alpha^{(t)})$  from the execution environment. The reward distribution over the  $(c^{(t)}, \alpha^{(t)})$  pair is considered unknown *a-priori*.

The CB problem can be considered a particularization of the full Reinforcement Learning (RL) problem, with the *context*  $c$  in CB analogous to the *state*  $s$  in RL. However, while the next context  $c^{(t+1)}$  in CB is independent of  $c^{(t)}$  and  $\alpha^{(t)}$ , in RL, the distribution of  $s^{(t+1)}$  depends both on the last state and on the performed action. This observation matches the introduced setup since the context, which includes the channel conditions and congestion factor, is not affected by the controller’s decision. Also, while reward estimation in RL requires accounting for future rewards using a discount factor  $\gamma$ , the reward in CB equals the instantaneous collected reward. Therefore, any CB problem can be solved using RL algorithms considering 1-step episodes or  $\gamma = 0$ .

ATHENA-ML uses an RL algorithm to solve the CB problem by adjusting it to 1-step episodes. The resulting policy function  $\pi(c) : \mathcal{C} \rightarrow \mathcal{A}$  is a deterministic function that maps the current context into action. The goal is to learn the optimal policy  $\pi^* = \arg \max_{\pi \in \Pi} [r(c, \pi(c))]$ , which maximizes the received reward. Additionally, ATHENA-ML, as another RL model, is model-free, in the sense that it does not have any insights about the environment’s internal model. On the contrary, it relies solely on the received reward signal, which renders ATHENA as an appealing solution for heterogeneous computational platforms.

## 4.2.1 ATHENA-ML Components

### 4.2.1.1 Context Space $\mathcal{C}$

Let  $c \in \mathcal{C}$  denote the system context, where the context space  $\mathcal{C}$  in ATHENA-ML incorporates the combination of (i) the congestion  $\beta \in \mathcal{B}$ , where  $\mathcal{B}$  is the set of eligible congestion factors of size  $|\mathcal{B}|$  and (ii) the UE's channel conditions  $\bar{c} \in \mathcal{R}$ , approximated via UE's SNR, discussed in more detail in Sec. 4.3. The first variable can be measured by the cloud provider metrics system, while the second is usually considered by state-of-the-art radio resource controllers. I define  $c^t := (\beta^t, \bar{c}_u^t)$  to be the context vector representing the system's context at stage  $t$ . The context space is  $\mathcal{C} := \mathcal{B} \times \mathcal{R}$ .

### 4.2.1.2 Action Space $\mathcal{A}$

As existing radio resource management works [83], ATHENA-ML takes two consecutive decisions: First, an assignment of a number of PRBs  $n_u \in \mathcal{N}$  for UE  $u$ . Second, a selection of an MCS over those PRBs, to match the UE channel conditions. Let  $m_u \in \mathcal{M} := \{1, \dots, M\}$  be the MCS decision for user  $u$ , where  $\mathcal{M}$  is a set of possible MCS in 5G.

Thus, the action space  $\mathcal{A} := \mathcal{N} \times \mathcal{M}$  essentially captures all possible pairs of decisions  $m_u^t$  and  $n_u^t$  allocated to user  $u \in \mathcal{C}$  in decision episode  $t$ . The decision/action is transferred to the user and modulates the UL frame accordingly. The environment is the UL processing computational platform and, more specifically, the LDPC decoder, which decodes the UL frame and gives back the reward signal.

### 4.2.1.3 Rescaling the Action Space for Scalability

In the configuration above, the size of the set  $\mathcal{A}$  increases linearly with the number of PRBs  $N$ . Especially for certain 5G deployments of 100 MHz and Subcarrier Spacing (SCS) 30 KHz,  $N$  equals 250, which is five times more than in the case of 10 MHz and SCS 15 KHz, showcasing the need for an efficient representation of the action space. To avoid this, the discrete nature of the decision variables  $\alpha_u^t = (n_u^t, m_u^t)$  can be transformed to the continuous one  $\hat{\alpha}_u^t = (\hat{n}_u^t, \hat{m}_u^t)$ . By doing so, the ML task gets simplified by making the action selection problem a continuous one, namely, a regression one. That is, the outcome variable is a continuous variable in the 2-D space, which can be discretized to space  $\mathcal{N} \times \mathcal{M}$ , according to a procedure described later. By moving to the continuous space, adjacent actions are inherently associated, which can not happen in the discrete space where every action is distinct.

#### 4.2.1.4 Reward Function

Given the  $\bar{c}_u$  condition for each user  $u$  and the congestion factor  $\beta$ , ATHENA encourages the actions  $\alpha$  that will probably lead to successful decoding, ultimately pushing the system to learn allocations that result in high system throughput<sup>1</sup>.

Successful decoding happens when (i) the data bits are transmitted without any errors under the SNR conditions, and (ii)  $T_{dec}$  does not exceed the deadline. Higher  $(n_u, m_u)$  combinations imply carrying more data with fewer redundant bits for error correction. Depending on the channel conditions and system congestion, this can lead to decoding failures after  $I_{MAX}$  iterations and deadline violations.

ATHENA-ML's reward function accounts for this observation. The reward captures the contribution of (i) the data bits  $d_u^t$  that user  $u$  is granted by the gNB to transmit under action  $\alpha_u^t$ ; (ii) a binary variable  $r_u^t$  denoting the result of the CRC of the TB after decoding it; (iii) the decoding time  $T_{dec,u}^t$  for the TB; and (iv) the target decoding deadline  $J$ . More specifically:

$$R = \begin{cases} d_u, & (T_{dec,u} \leq J - 1) \text{ and } (r_u = 1) \\ -K, & \text{otherwise} \end{cases},$$

where  $K$  is a positive constant term for penalizing wrong decisions. In Sec. 4.4, it shows that ATHENA-ML learns quickly, because it takes advantage of both CRCs' values and decoding time.

## 4.2.2 ATHENA-ML Internal Design

### 4.2.2.1 Actor-Critic Design

ATHENA-ML is solving the CB problem using a 1-step episode RL architecture that follows the Actor-Critic (AC) paradigm [84], which has shown superior scalability properties [84, Chapter 13.1]. AC belongs to the policy gradient family of algorithms, can operate on continuous-valued control spaces, and approximate directly the best policy function  $\pi$  across the reward  $r$  obtained by each state-action pair, and thus its direct applicability to the 5G NR systems, especially the one based on O-RAN. On the other hand, value-based algorithms (*e.g.*, SARSA, Q-Learning, DQN) operate on distinct control spaces, approximate the state-value or the action-value function, and then apply an  $\epsilon$ -greedy selection policy on the control space. This becomes intractable in big action spaces, such as in large bandwidth 5G deployments.

Deep Deterministic Policy Gradient (DDPG) [85] algorithm has stood out due to its

---

<sup>1</sup>This assumption makes ATHENA-ML better suited for Enhanced Mobile Broadband (eMBB) scenarios, ignoring other metrics that could be useful for *e.g.*, low latency scenarios such as the guarantees on the maximum length of UE transmission queues.

benefit in resolving deterministic continuous action problems. The DDPG agent comprises two functions: the *actor* and the *critic*. The critic approximates the action value function  $Q(c, \alpha)$  that predicts the expected reward, which is received by the environment when performing action  $\alpha$  in context  $c$ . The critic is represented by a neural network  $Q_\phi(c, \alpha)$ , parameterized by weights  $\phi$ . Given a set of interactions  $\mathcal{D}$  consisting of samples  $(c, \alpha, r)$ , it minimizes the residual Mean Squared Error (MSE) between the predicted  $Q_\phi(c, \alpha)$  and the received reward  $r$ :

$$\mathbb{E}_{(c, \alpha, r) \sim \mathcal{D}} [(Q_\phi(c, \alpha) - r)^2] \quad (4.1)$$

The actor approximates the policy function. It is represented by a neural network  $\mu_\theta$ , with weights  $\theta$ , and gives the deterministic action  $\hat{\alpha} = \mu_\theta(c)$ . The goal of the actor is to output the action  $\hat{\alpha}$  that maximizes  $Q_\phi(c, \hat{\alpha})$ :

$$\mathbb{E}_{c \sim \mathcal{D}} [Q_\phi(c, \mu_\theta(c))] \quad (4.2)$$

#### 4.2.2.2 Actor Inference

For scalability reasons, the output action rescales so that the actor gives the approximate continuous  $\hat{\alpha} = (\hat{n}, \hat{m})$  instead of the discrete  $\alpha = (n, m)$ . In order to discretize  $\hat{\alpha}$ , the following hierarchical blocks were implemented [86]:

1. **Action Generation:** Action generation uses the K-Nearest Neighbors (k-NN) function  $g_k(\hat{\alpha}) = \arg \min_{\alpha \in \mathcal{A}}^k |\alpha - \hat{\alpha}|_2$ , where  $k$  denotes that the  $k$  actions in  $\mathcal{A}$  that are closest to  $\hat{\alpha}$  by  $L_2$  distance.
2. **Action Refinement:** Even though the actions are close to each other in  $\mathcal{A}$ , they may have a complete direct impact on the environment, and blindly choosing the closest to  $\hat{\alpha}$  is not ideal. For example, in low SNR conditions, if the MCS component  $\hat{m}$  of the output of the actor is 9.8 then selecting the closest  $m = 10$ , which applies 16-QAM modulation, instead of  $m = 9$ , which applies QPSK modulation and therefore severely lower complexity, may have a detrimental effect on the decodability of the frame. Hence, each of the  $k$  actions is considered, and only the highest rewarding one according to the prediction of the critic is eventually selected according to:

$$\pi_{\theta, \phi}(c) = \underset{\alpha \in g_k \circ \mu_\theta(c)}{\operatorname{argmax}} Q_\phi(c, \alpha) \quad (4.3)$$

### 4.2.2.3 Application to O-RAN and 5G-NR Standards

As depicted in Fig. 4.2, ATHENA-ML acts on several components of the 5G NR and O-RAN architectures. Indeed, ATHENA-ML acts in the MAC layer of the 5G NR stack, supporting the decisions of the scheduler at every slot. The design is such in order to cope with fast-changing conditions on the radio channels since speeds up to 120 km/h and 500 km/h for vehicles and high-speed vehicles, respectively, have to be supported, according to the 5G standard [87]. Frequent user scheduling, at every slot to provide high throughput and low latency, and high 5G high bands incur high SNR variations within very few slots, which is discussed later. This requires rapid reactions from the gNB side to maintain high throughput, by achieving decodability and staying within the deadline constraints. ATHENA-ML is placed in the MAC layer of the O-DU directly in inference, hence being able to support very fast responses from the model, while the training can be performed in the Near Real-time RAN Intelligent Controller (Near-RT RIC), decoupled from the user plane.

It is worth noting that, as shown in Sec. 4.4, the proposed ATHENA-ML framework was not explicitly evaluated at these speeds. Such scenarios are characterized by rapid fluctuations of the instantaneous SNR, which pose a challenge for link adaptation mechanisms. However, ATHENA-ML is able to address these dynamics since it performs inference at slot-level granularity. This fine-grained operation allows the scheduler to promptly react to channel variations even at very high mobility. By contrast, alternative approaches such as `vrain` [63] operate on coarser time scales (on the order of 1-10 seconds), typically by fixing a maximum MCS. Consequently, these schemes lack the reactivity required in fast-varying channels, whereas ATHENA-ML is inherently designed to cope with such conditions.

For episode  $i$  at slot  $t$ , the HARQ process (residing in the MAC layer) queries ATHENA-ML with the context  $c_i$  and receives the controller's decision  $\alpha_i$ . The action is enforced in the PUSCH channel at slot  $t + J$ , and the reward from the LDPC decoder  $r_i$  is collected at slot  $t + 2 \cdot J$ . Because of the nature of the HARQ processes, a single acting agent would suffer from the *delayed reward* problem; at slot  $t + 1$  the agent has to take an action for the episode  $i + 1$  before the reward of the  $i$ -th episode is collected.

To leverage the independent lifecycle of each HARQ's data transmission, the DDPG agent is demultiplexed into  $|H|$  agents, as many as the HARQ processes. Each HARQ agent shares the same parameters  $\theta, \phi$ . Each agent  $i$  interacts with the HARQ process  $i$ . At each lot  $t$ , the agent indexed by  $t \bmod |H|$  is designated to handle the transmission. Following the same procedure, the decoder returns the reward to the corresponding agent. Since the HARQ agents interact with the HARQ (MAC) and the decoder (PHY), which require fine time resolution, they fit in the O-DU.

The HARQ agents offload the learning function to the *main* agent. The main agent shares the same model parameters as the HARQ agents. At every scheduling opportunity,

the HARQ agents store the samples, consisting of context, action, and reward tuples  $(c, \alpha, r)$  in their internal buffer. At periodic timings, the main agent collects these samples over the E2 O-RAN interface, calculates the gradients, optimizes the model, and pushes back the updated weights. Since the sample collection does not have strict timing constraints, the main agent can reside either on Near-RT RIC or Non Real-time RAN Intelligent Controller (Non-RT RIC). In Fig. 4.2, I show how ATHENA-ML integrates with the O-RAN architecture. The main agent is running as an xApp in the Near-RT RIC, and the HARQ agents running in O-DU as *dApp*, adopting the concept from [88].

#### 4.2.2.4 Offline Training

O-RAN principles require *pretraining* of the agent models before they are deployed on a live production environment [89]. To comply with this requirement, I extensively collected a dataset  $\mathcal{D}$  from an isolated sandboxed environment multiple samples for randomly taken decisions  $\alpha_t = (n_t, m_t)$  and contexts  $c_t = (\beta_t, \bar{c}_t)$ , and recorded the decoding time  $T_{dec}$  and the CRC result  $r$  as returned by the LDPC decoder. To accelerate training, reduce the sample complexity, and remove unavoidable outliers produced by a real system, the dataset is grouped per  $(\beta, \bar{c}, n, m)$ , and the returned reward is modified for each group as follows:

$$R^{group} = \begin{cases} d_u, & P[r = 1] \geq r_{thres} \text{ and } T_{dec}^\gamma \leq J - 1 \\ -K, & \text{otherwise} \end{cases} \quad (4.4)$$

where  $T_{dec}^\gamma$  is the  $\gamma$ -percentile  $\in [0, 1]$  of  $T_{dec}$  within the group and can be considered as a proxy for *performance vs reliability trade-off*. Due to the stochastic nature of the computing platform, there is high variability of the decoding times, which can be attributed to measurement imperfections, incapability to isolate the low-level caches of the processors, virtual memory invalidation during context switches *etc.* Picking higher  $\gamma$  would lead to learning more conservative policies, because of the negative reward, and vice-versa.  $r_{thres}$  is a target threshold value above which satisfactory data transmission can take place, in terms of successful error correction and wireless performance, *i.e.*, lower Block Error Rate (BLER).

## 4.3 Implementation

To prove the feasibility of ATHENA, I implemented and integrated it with srsRAN [35] testbed, described in Chapter 3. ATHENA was implemented in a compliant way with the O-RAN reference architecture. For the implementation of ATHENA-ML<sup>2</sup>, however, I did

---

<sup>2</sup>All the software components related to ATHENA-ML are available at [https://github.com/kaposnick/athena\\_agent](https://github.com/kaposnick/athena_agent)

not directly rely on this default implementation of `srsRAN`, for reasons mentioned below.

#### 4.3.1 ATHENA-ML Integration into vRAN Software

From Sec. 4.1, the two main variables that influence the decoding time of a frame are (i) its MCS index  $m_u$  and the number of PRBs it spans  $n_u$  and (ii) the intrinsic complexity of the LLR maximization operation, which depends on the selected MCS and the SNR.

To be able to train ATHENA-ML against very different conditions, and hence allow the algorithm to cope with dynamic scenarios, I needed to span over very different channel conditions. The `GRC Broker` is the component of the testbed that adjusts the perception of the signal strength. The multiplier boxes between UE and cells modified the transmitted samples by a constant gain  $G$  value in the range  $[.05, 1]$ . When  $G = 1$ , the signal is received at full strength, measured at 30 dB per PRB at the gNB side, while when  $G = .05$ , the SNR drops to 5 dB, below which the gNB can hardly decode any control or data channels. In the experiments, the broker dynamically controls  $G$  between .05 and 1 to emulate different channel conditions. Finally, to simulate an AWGN channel, I added an additive noise process to the intercepted symbols, which samples from a normal distribution with zero mean.

ATHENA-ML is implemented using `Python` and `Tensorflow`. ATHENA-ML directly integrates with `srsRAN` scheduler, by overriding the default controller and making the decision about what  $(n_u, m_u)$  to assign to the different users. The context space variables are available to ATHENA-ML either directly, as in the case of the SNR, or indirectly, as the  $\beta$  factor is sent to the controller by the monitoring application. ATHENA-ML has one *coordinator* process, responsible for directing requests and the decoding results to the corresponding worker agent based on the slot,  $|H| = 8$  HARQ agents' processes, as many as the HARQ processes. Since no online training is taking place on the experiments, the *main* agent was not deployed in the Near-RT RIC. The communication between `srsRAN` and ATHENA-ML is achieved using *Linux named pipes* and between the coordinator process and the HARQ agents using *shared memory buffers*.

The factor  $\beta$  models all the possible interfering factors for the UL frame decoding times. Since this implies an additional delay,  $\beta$  is implemented by introducing an additional workload at the end of each TB decoding iteration: For each  $\beta$ ,  $\beta_n = 1000 \cdot \beta$  square root computations during UL frame decoding are executed, essentially reducing the effective capacity offered by the cloud platform.

#### 4.3.2 ATHENA-ML Pipeline

In order to train and execute ATHENA-ML, several factors need to be addressed, including the gathering of the training data and the machine learning operation.

### 4.3.2.1 Dataset Collection

To collect the training dataset  $\mathcal{D}$ , I replaced the default `srsRAN` controller with a custom one that randomly selects a  $(n, m)$  decision and assigns an UL grant to the user. To span across different  $\beta$  and  $\bar{c}$ , there is an automated process that sets the gain  $G$  of the channel and the congestion conditions on which the decoder threads are running. I deployed the end-to-end cloud-native RAN testbed described in Chapter 3. The UE is configured to transmit uplink `iperf` traffic, while at each slot the custom scheduler assigns a scheduling grant  $(n, m)$ . Upon processing the corresponding PUSCH, the gNB records the decoding latency  $d$  (in ms) and the binary decoding outcome  $r$  (OK/KO). For every slot, the gNB stores in a CSV file the tuple:  $(\beta, \bar{c}, n, m, d, r)$ . To ensure that the decoding threads are not preempted by the Linux kernel, I bind them to a dedicated set of CPUs using the `cpuset` utility and further isolate these CPUs with the `isolcpus` option. This configuration prevents the default Linux scheduler from interrupting the decoder threads in favor of other processes.

### 4.3.2.2 Actor Critic Internal Structure

The actor’s neural network is implemented using fully connected layers with ReLU activation. It has 2 output neurons with sigmoid activations, denoting the  $(\hat{n}, \hat{m})$ , which discretize to  $(n, m)$  using Eq. 4.3. The critic’s neural network also comprises fully connected layers with ReLU activation and one single output neuron, outputting the reward prediction, with linear activation.

### 4.3.2.3 Pretraining

Given the dataset  $\mathcal{D}$ , I grouped according to the procedure in Sec. 4.2.2.4. First, I pretrained the critic’s neural network  $Q_\phi$  as a normal regressor that minimizes the critic’s objective, using Eq. 4.1. Successively, I froze the critic’s weights and pretrained the actor’s neural network  $\mu_\theta$  so that it maximizes its objective, *i.e.*, the critic’s reward prediction, using Eq. 4.2. I pretrained both actor and critic networks using the Adam optimizer, with a learning rate  $1\epsilon^{-4}$  for a period of 200 epochs. The collected dataset was divided into 80% for training, 10% for validation and 10% for testing purposes. In turn, the agent’s weights  $\theta, \phi$  on `ATHENA-ML` are deployed on the described testbed, and the agent is inferred using Eq. 4.3 using  $k = 5$ .

### 4.3.2.4 Scalability Enhancements

While the `srsRAN`’s `ZeroMQ` RF-frontend driver allows to test `ATHENA` in a real system, its capabilities are mostly targeting the debugging of the higher layers of the RAN stack. Hence, one of these limitations is the native support for just one UE per gNB cell.

While a possible solution to this issue could have been an enhanced software-based channel emulator using the already integrated GNU Radio module, to overcome this and simulate a multi-user cell scenario, I designed a *digital twin*. A digital twin is a replica of a physical system that can replicate the system's environment, eliminating the need to repeat exhaustive physical tests. In that case, the DT generates the same data distributions as the physical vRAN's LDPC decoder, whose output affects ATHENA-ML's decisions. Multiple users can now *infer* the digital twin, eliminating the restrictions of srsRAN's front-end. More details on the digital twin are given in Chapter 5.

#### 4.3.2.5 Traffic Shape

ATHENA-ML has been trained to control scheduled users with full upload buffer, *i.e.*, to serve their maximum achievable throughput, which in ideal channel conditions is achieved at maximum PRB and MCS. In this sense, ATHENA provides the maximum TBS that can be decoded within the deadline. In order to avoid excessive redundant bits in case the user's demand is less than the predicted maximum number of bits, I adjust ATHENA-ML's decision so that the equivalent TBS fills the requested demand. I also keep the new MCS less than ATHENA-ML's decided MCS to maintain the decodability of the frame.

#### 4.3.2.6 SNR Estimation

The broker adopts the universal channel model:  $y = Gx + n$ , where  $x, y$  are the input and the output of the wireless channel, respectively,  $G$  is the channel's impulse response and  $n \sim N(0, \sigma)$  is the noise that follows a normal distribution with zero mean. SrsRAN's eNB/gNB implementation, by default, computes the SNR using the exponentially weighted average of the instantaneous SNR in the PUSCH channel. The instantaneous SNR is estimated as the average power per PRB divided by the noise estimate, which is flat across the frequency spectrum. However, in wideband 5G deployments (spanning up to 100 MHz), the power per PRB may fluctuate because the bandwidth of the system has a higher probability of crossing with the coherence bandwidth of the channel, converting the channel response  $G$  from flat-fading to frequency-selective fading. To avoid specifying the location and the size of allocated PRBs with higher SNR (which are further restricted by Resource Block Group parameters [36, Chapter 6.1.2.2]), the channel condition input variable acts conservatively and takes the minimum power as reference for the calculation of the instantaneous SNR.

On the other hand, srsUE, srsRAN's implementation of user equipment, distributes uniformly the power per PRB so that the total transmitted power per subframe (1 ms) falls under a certain threshold [90, Chapter 6.2]. This observation is crucial, since for the same channel response  $G$  and following the averaging procedure described above, a higher number of PRB yields lower instantaneous SNR and vice versa.

Considering the SNR as a proxy for the channel conditions, the channel condition can be estimated:

$$\bar{c} = SNR + 10 \cdot \log(PRB) \quad (4.5)$$

where  $\bar{c}$ ,  $SNR$  are expressed in dB. The variable  $\bar{c}$  effectively measures the total SNR, and Eq. 4.5 yields the same  $\bar{c}$  for the same channel response  $G$  irrespective of the number of PRBs.

#### 4.3.2.7 CPU Congestion Factor $\beta$

O-Cloud allows for the parallel execution of heterogeneous jobs. Private and public cloud providers offer the tenants the capability to select certain filters on the physical servers (*e.g.*, availability zone, hardware acceleration), but, in principle, tenants are agnostic of the actual silicon their application is running on. The CPU congestion factor  $\beta$ , which effectively expresses the number of available CPU cycles that are allocated to the applications, can be directly configured by popular cloud orchestrators (such as `Kubernetes`) or specific orchestration algorithms, such as `ATHENA-MR`, detailed in Chapter 5. Additionally, O-Cloud is enhanced with monitoring applications (*e.g.*, `Prometheus` [91]), which constantly track alerts and measure the resource consumption (CPU congestion, memory, network/disk IO, *etc.*) of the individual jobs, servers, or whole infrastructure. The monitoring application runs on the Service Management and Orchestration Framework (SMO) of the O-RAN architecture in the Non-RT RIC. Independent of whether  $\beta$  is directly configured or measured, its value is available at the SMO level and can be conveyed to infrastructure-aware applications to provide them with the current status of resource usage. `ATHENA-ML`, as a CPU-aware vRAN application running on O-DU, receives as input the current CPU congestion metric from SMO and adjusts its scheduling decisions concerning the infrastructure contextual fluctuations. In Fig. 4.2, SMO monitors the O-Cloud infrastructure via the O2 interface and informs `ATHENA-ML` via the O1 interface.

#### 4.3.3 Multi-User Scenario

`ATHENA-ML` radio resource controller works on a single-user basis, allowing for selection of the optimal MCS and PRB combination at every point in time. `ATHENA` can integrate with different UE selection procedures. I picked `srsRAN`'s round-robin and matched it to `ATHENA-ML` that selects the best PHY layer parameter. Hence, for each slot, a single user is circularly selected and scheduled, and their SNR is given as input to `ATHENA-ML`, which then controls their MCS and number of PRBs. The training data for this scenario is gathered using the digital twin. The integration of `ATHENA-ML` with the round-robin

user selection procedure imposes a maximum boundary on the number of selected users at each slot and hence may not be suitable for scenarios that are more latency-constrained. Other UE selection algorithms may be implemented leveraging the ATHENA-ML controller, exploiting the models and interfaces that have already been discussed.

## 4.4 Evaluation

In this section, I evaluate ATHENA-ML against the vanilla radio resource controller available in srsRAN, namely **Baseline**, which serves as a general benchmark for commercial state-of-the-art radio resource managers. Among the 3GPP-compliant CPU-aware MAC schedulers that provide experimental evaluation referenced in the background chapter 2, **vrain** [63] and **CloudRIC** [8] are the most closely related to this work. As outlined in Sec. 4.2, however, **vrain** operates at a coarser granularity and does not make slot-level decisions. In contrast, **CloudRIC** is not CPU-only but rather considers both CPU and GPU resources, and also did not offer full integration with a mobile stack such as srsRAN. To demonstrate the need for having a joint PRB-MCS selection, I have also trained, deployed, and evaluated an alternate version of ATHENA-ML, namely **ATHENA-MCS**. **ATHENA-MCS** is a variation of the ATHENA-ML algorithm that follows the same actor-critic architecture but only controls the applicable MCS, leaving the number of PRBs set as the maximum. Both ATHENA-ML and ATHENA-MCS interact with the UE scheduler solution, matching the available capacity to the contextual condition, showing how the ATHENA-ML framework can be leveraged by different UE scheduling algorithms. For the experiments, I used an Intel Xeon Gold 6240R CPU server with 16 cores, hyperthreading disabled, equipped with 64 GB of RAM.

### 4.4.1 ATHENA-ML Convergence

In Fig. 4.3, the actor’s objective, *i.e.*, maximizing the critic’s average reward prediction, is depicted. I trained the two versions of ATHENA-ML for different reliability values  $\gamma \in [50, 90, 99]\%$  and different decoding deadlines  $J \in [3, 4]$  ms. I set  $r_{thres} = 90\%$ , since O-RAN defines 10% maximum BLER, and penalty factor  $K = 1$ . Both versions of ATHENA-ML learn to, indeed, pick high reward combinations of  $(n, m)$  in higher deadlines, leading to greater collected rewards, since there is greater available time to decode bigger packets. Higher  $\gamma$  drives the agent to learn conservative policies, *i.e.*, less performant combinations of  $(n, m)$ , since a higher value of decoding times is considered as a target and is compared against the decoding deadline for the reward evaluation in Eq. 4.4. Conversely, lower  $\gamma$  implies more aggressive policy learning, which, though in the production system in inference mode it can have a detrimental effect due to its lower reliability. Note, also, that ATHENA-ML performs better in terms of the average actor’s objective than ATHENA-MCS, attributed to the bigger and finer action space (more details

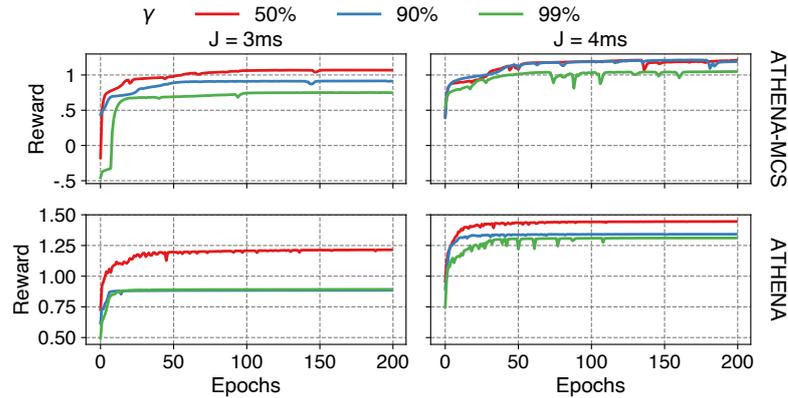


Figure 4.3: Actor’s objective across training epochs for reliability levels  $\gamma = [50\%, 90\%, 99\%]$  and for the two variants; **ATHENA-ML** (labelled as *ATHENA*) and *ATHENA-MCS*.

in Sec. 4.4.2).

Through the rest of the evaluation section, unless otherwise stated,  $J = 4$  ms is used, which is the typical eMBB case in 5G [8] (and 4G, where this value is fixed [77]), and  $\gamma = 99\%$  to ensure high reliability.

#### 4.4.2 ATHENA-ML Performance

Next, **ATHENA-ML** is compared against **Baseline**. It implements a round-robin UE selection policy, and for PRB selection, it assigns the maximum number of PRBs in case of a single UE. For MCS selection, it maps the SNR to a target code rate, via a proprietary SNR-to-CQI table [92] and the 3GPP CQI-to-coderate table [36], and through an MCS-TBS loop, it picks the minimum MCS index that yields the coderate closest to the target one. I evaluate **Baseline** and **ATHENA-ML**, trained with  $J = 4$  and  $\gamma = .99$ , in a single UE scenario that transmits uplink UDP data at full buffer speed.

For the experiments, using a slot of 1 ms (numerology  $\mu = 0$ ) and 10 MHz of bandwidth, it gives 50 PRBs in the UL, out of which  $N = 45$  are only available for UL data transmission. To compare the success of the solution during the first transmissions of the frame, **srsRAN** was modified to disable HARQ retransmissions, *i.e.*, using a single redundancy version. The GRC broker adjusts the channel gain  $G$  of the UE so that the perceived SNR per PRB at the gNB side varies between 5 dB and 30 dB.

In Fig. 4.4, the solid lines depict the average throughput (in Mbps) achieved by **ATHENA-ML**, **ATHENA-MCS**, and **Baseline** at each SNR level. The throughput is calculated as the total number of bytes correctly decoded within the deadline, divided by the number of slots processed during the simulation. A TB is considered successfully decoded when  $T_{dec}$  is below 3 ms and has a successful CRC check. When  $\beta = 0$ , where neither controller suffers losses from deadline violations, **Baseline** substantially

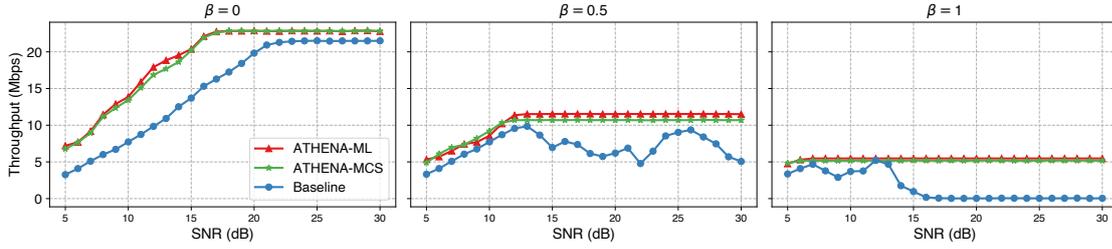


Figure 4.4: Throughput served by a gNB running ATHENA-ML, ATHENA-MCS and Baseline algorithm, for different congestions  $\beta$  and SNR between  $[5, 30]$  dB.

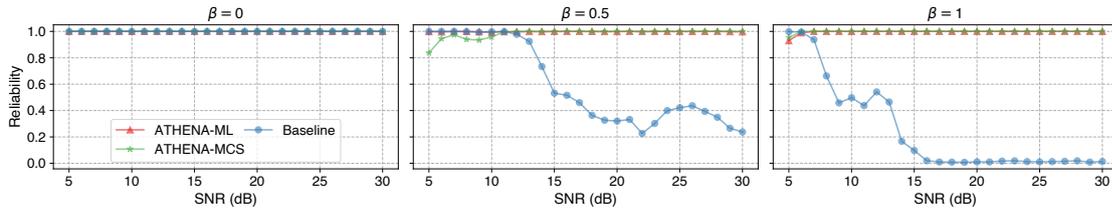


Figure 4.5: Reliability by a gNB running ATHENA-ML, ATHENA-MCS and Baseline algorithm, for different congestions  $\beta$  and SNR between  $[5, 30]$  dB.

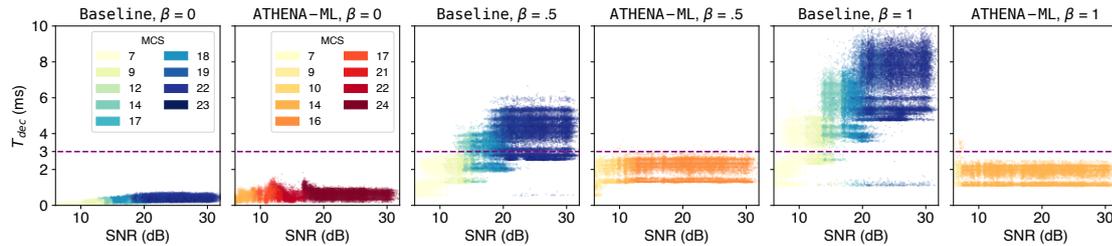


Figure 4.6: The  $T_{dec}$  distribution and the selected MCS, for ATHENA-ML and Baseline algorithm, with different congestion  $\beta$ .

underperforms both ATHENA-ML and ATHENA-MCS, in terms of throughput. ATHENA-ML records an improvement of 3.94 Mbps with a maximum improvement of 7.45 Mbps at 13 dB. This originates from the conservative nature of traditional controllers, where they stick to simulation-based min-MCS approaches to preserve the decodability of the frames, while the data-driven approaches can discover optimal control policies (*e.g.*, at high SNR ATHENA-ML picks MCS 24, while Baseline goes up to 23).

Higher  $\beta = .5$  demonstrates similar behavior, though the congestion factor has an important impact on the decoding time. Baseline's throughput drops due to low reliability, as depicted in Fig. 4.5, where it counts the percentage of the frames where either  $CRC = 0$  or  $T_{dec} \geq 3$ . The maximum achievable throughput for both versions of ATHENA-ML drops to 11.5 Mbps with very high-reliability levels.

When  $\beta = 1$ , Baseline's reliability essentially drops to 0, along with the throughput.

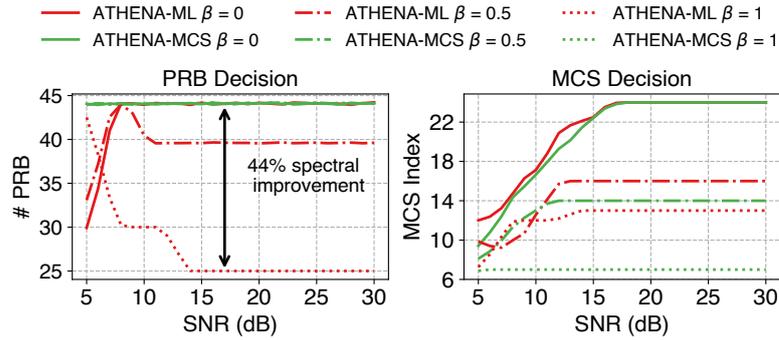


Figure 4.7: MCS and PRB decisions of the two versions ATHENA-ML and ATHENA-MCS.

For ATHENA-ML, the SNR has minimum impact, because the high congestion forces the agent to apply a single control policy for all the channel conditions.

Fig. 4.6 shows the decoding time  $T_{dec}$  yielded by the two approaches. While in  $\beta = 0$ , all  $T_{dec}$  are below 3 ms, the variability is much higher in ATHENA-ML, which is explained by higher MCS selection. Again, with higher  $\beta$ , a non-negligible part of the frames misses the deadline. ATHENA-ML, instead, lowers either the MCS or the PRB to always meet it. As a result, it obtains a flatter  $T_{dec}$  distribution, which is a direct consequence of the selected actions.

#### 4.4.3 Advantages of Joint MCS-PRB Management

I now compare the two alternate versions of ATHENA-ML and ATHENA-MCS. Fig. 4.7 shows the PRB and MCS decisions of the two versions for different SNR levels and three congestion factors. While ATHENA-MCS always transmits at full bandwidth, ATHENA-ML explores different alternatives in the 2D action space. As observed in Fig. 4.4, both versions manage to achieve optimal levels of reliability and equal performance, with ATHENA-ML slightly oversteering by 500 Kbps in medium congestion factors, transmitting at lower PRBs, and increasing the MCS. The major advantage of ATHENA-ML over ATHENA-MCS is the up to 44% spectral efficiency improvement (in terms of PRB utilization) since it can achieve equal or slightly better performance using less bandwidth, leaving available space for other users to be scheduled. This aspect could be leveraged by a different UE selection procedure, capable of integrating multiple UEs in the same slot, that jointly selects users and decides MCS and PRB, so unused PRBs may be re-assigned to other UEs to increase the total throughput of the system, respecting the time budget as well as technology requirements and user characteristics. Note that both the original ATHENA-ML algorithm and ATHENA-MCS are based on the same original architecture and should be considered variants and not alternative solutions.

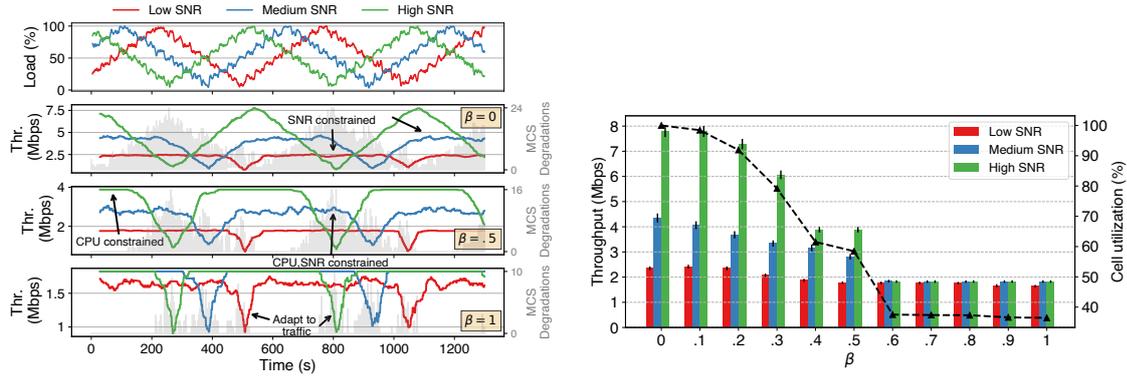


Figure 4.8: Left: Throughput for different congestion factor  $\beta$  and user load. Right: Throughput for different congestion factor  $\beta$  and cell utilization.

#### 4.4.4 Multi-User Scenario

As explained in Sec. 4.3.3, the digital twin can serve to evaluate the performance of the proposed architecture in a multi-user setting. Although the setup was tested with more users, in this experiment I only consider 3 users in the UL for readability, corresponding to 6 db (*low*), 13 db (*medium*), and 20 db (*high*) SNR with a standard deviation of 3 dB. At each slot, a user is selected in a round-robin fashion. Given the user's channel conditions and congestion factor, the pretrained ATHENA-ML agent provides the control decision. Subsequently, the digital twin is inferred, outputting the expected decoding time and the decoding success probability. The UL frame is considered decoded if the predicted decoding time is less than the deadline and the sampled success probability process is 1.

To capture the capability of ATHENA-ML to adapt to different UL traffic shapes, the UEs' load is artificially modified (left plot and first row of Fig. 4.8) in 1300 seconds. This load reflects the demanded bits to be transferred at the UE's MAC level and is conveyed in the BSR message to the base station. Accordingly, in the following three rows, the throughput is modified for each user. The throughput follows the pattern of the traffic load. ATHENA-ML achieves this by performing MCS degradations, which are shown in the shaded gray area.

In the right plot of Fig. 4.8, the same experiment is repeated but for full traffic load (100%). The bars depict the mean throughput, and with error lines the jitter of each user for variable  $\beta$  values. In low  $\beta$  and high SNR scenarios, ATHENA-ML yields high jitter (due to the SNR fluctuations), which gets lower either when the SNR drops or the congestion goes up. For  $\beta \geq .6$ , the SNR is almost taken into no account since the congestion impact supersedes. In the same plot, the cell utilization, defined as the total throughput of the users divided by the throughput at  $\beta = 0$ , denotes the maximum achievable throughput for these channel conditions. A gradual drop of cell utilization in low congestion factors

occurs, reaching 60% in medium  $\beta$ , before falling to below 40% in high  $\beta$ , showcasing  $\beta$ 's impact in the model's predictions.

## Summary

In this chapter, I presented the design, implementation, and evaluation of **ATHENA**, a CPU-aware MAC scheduler for reliable O-DU deployments in resource-constrained O-RAN cloud infrastructures. **ATHENA** is an RL-based MAC resource controller that dynamically allocates the MCS and PRBs to UL users at each slot, adapting to the congestion level in the O-DU to ensure that uplink processing pipelines are reliably met. Within the O-RAN framework, **ATHENA** is deployed as a dApp in the Near-RT RIC.

Through experimental evaluation on a 3GPP-compliant testbed and benchmarking against the baseline scheduler, **ATHENA** consistently achieves high reliability levels under high-congestion scenarios, where the baseline scheduler's reliability and throughput collapse to zero, which demonstrates its practical capability for seamless integration in standards-compliant deployments.



# 5

## Explainability and Twinning for AI on RAN

---

### Disclaimer

This chapter is based on the following publications:

- **Nikolaos Apostolakis**, Marco Gramaglia, Livia Elena Chatzieftheriou, Tejas Subramanya, Albert Banchs and Henning Sanneck, "ATHENA: Machine Learning and Reasoning for Radio Resources Scheduling in vRAN Systems," in *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 2, pp. 263-279, Feb. 2024, doi: 10.1109/JSAC.2023.3336155
- **Nikolaos Apostolakis**, Livia Elena Chatzieftheriou, Dario Bega, Marco Gramaglia and Albert Banchs, "Digital Twins for Next-Generation Mobile Networks: Applications and Solutions," in *IEEE Communications Magazine*, vol. 61, no. 11, pp. 80-86, November 2023, doi: 10.1109/MCOM.001.2200854

Minor changes in structure and notation have been made to improve readability and consistency within this thesis.

This chapter describes two complementary aspects that enhance the design and deployment of ATHENA, Digital Twin (DT), and Explainable AI (XAI) within ATHENA's model. First, Sec. 5.1 introduces the DT designed to replicate the behavior of the LDPC decoder within the ATHENA framework. This DT enabled the fast evaluation and assessment of different RL architectures by mimicking the real-system behavior of the environment of the use case. The second part (Sec. 5.2) discovers the underlying patterns that drive the internal operation of the actor-critic functionality of ATHENA. I unveil the three kinds of explanations drawn from the deployed agent, namely *attributive*, *contrastive*, and *actionable*, and provide a use-case where the latter can be used in order to trigger reorchestration of O-RAN Cloud (O-Cloud) compute resources, *i.e.*, the cloud infrastructure hardware resources that host the O-DUs software and can be reorchestrated

by the SMO through the O2 interface.

## 5.1 Digital Twin

The design of an ATHENA entailed the electing, training, and deploying Deep Learning (DL) models, which can be challenging given the complexity of a real testbed, described in Chapter 3. The operations are happening at different levels of the stack, *e.g.*, grant assignment decisions are made in the MAC layer, the input about the CPU resources comes from the SMO, and the decoder performance is recorded at the PHY layer.

A second challenge is the high complexity of carrying out the training. In order to learn a good model, the AI/ML needs to visit all the possible input combinations several times. This may turn into a very long operation in a real system, as *(i)* real scheduling decisions are limited to once every slot, and *(ii)* it may be difficult to reach certain combinations of the inputs, due to channel conditions. This cannot be performed on the fly in a production system, as for example, O-RAN forces the deployment of offline pre-trained models [89]. These aspects make it impractical to learn from a real system, as a number of aspects, such as model architectures and hyperparameter configurations, analysis, and model interpretability, have to be taken into account. To overcome these issues, I designed the network DT of an environment that ATHENA operated, *i.e.*, the LDPC decoder functionality, that can improve the operation of training computationally-aware MAC schedulers.

### 5.1.1 Design

In ATHENA's use case, the vRAN functional block that has to be controlled using RL is the LDPC decoder, since it provides feedback on both the decoding time and the CRC result. Therefore, the network DT of such a system must replicate the physical system's distribution of decoding times and decoding success and failures. I created a dataset with real-trace performance measurements and then trained a supervised model to capture them in the DT.

The dataset contains different combinations of the input space, namely the CPU Capacity (in % of maximum CPU resources), the user's SNR, the MCS index, and the number of PRBs. The decoding result is represented by a binary variable, *i.e.*,  $CRC = 1$  if the frame is successfully decoded, or  $CRC = 0$  otherwise, and the decoding time is modelled as a continuous normally distributed variable, truncated to positive values. This choice is motivated by the empirical observation that decoding times are roughly symmetric around a mean, with limited extreme values, which is well captured by a Gaussian distribution. The DT outputs *i)* the probability with which a frame is successfully decoded, and *ii)* the mean and standard deviation of the normal distribution that models the decoding time. I measured the wall time taken by the decoder to finish

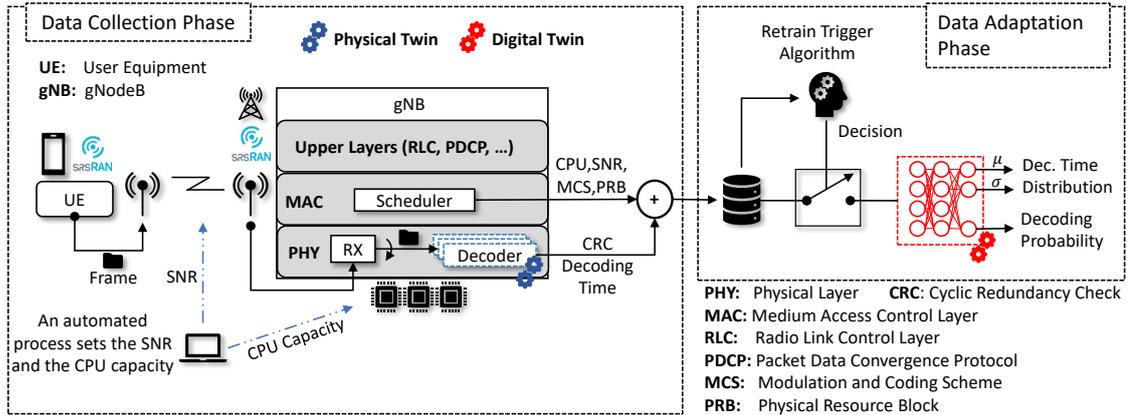


Figure 5.1: The building blocks of the proposed DT of ATHENA.

the task, and to eliminate the noise from other processes, I pinned the decoder threads to a specific CPU set and prevented the default Linux scheduler from preempting those threads in favor of other jobs. The CRC result was directly retrieved by the decoder. I collected  $\approx 14$  million samples, with extensive combinations of the input parameters. The *data collection phase* is shown in the left block of Fig. 5.1.

### 5.1.2 Model

The decoder's DT is represented by a two-headed Neural Networks (NNs) with common hidden feed-forward fully-connected layers. The last common hidden layer is divided into two branches, one for the decoding time prediction task and another for the decoding success prediction task. The last layer comprises three independent neurons; the decoding time's mean and standard deviation, and the probability of successful decoding. Their activation functions are linear, soft-plus (to ensure positivity), and sigmoid (to produce values between 0 and 1), respectively.

Similarly, the different tasks contain different activation functions. The prediction of the decoding successes uses the Binary Cross Entropy (BCE) loss, which is the negative of the log of corrected predicted probabilities and is standard for binary classification tasks [93]. The prediction of the mean and standard deviation parameters of the normal distribution uses the Negative Log-Likelihood (NLL) loss function, consistent with probabilistic regression of Gaussian distributions [94].

DT's NN was trained using backpropagation and the Adam optimizer with a learning rate of  $10^{-4}$ . For training and evaluation, the original dataset was partitioned into 80% for training, 10% for validation, and 10% for testing. The training of the DT with new samples is controlled by a *Retrain Trigger Algorithm*, which is fed by the newly collected samples and decides whether to trigger new retraining or not. As explained in Sec. 5.1.3, a simple heuristic that computes a distribution similarity of the new samples against the old ones can be used to trigger retraining if this metric drops below a certain threshold. In general,

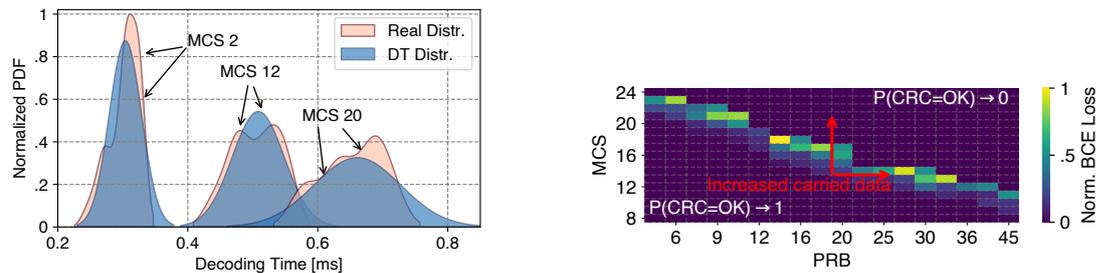


Figure 5.2: Left: PDF of the real and predicted decoding times. Right: Decoding probability prediction and BCE loss.

though, more complex re-training procedures may be used. Alternative algorithms could depend on the amount of newly collected samples or occur on a fixed frequency. For NN-backed DTs, identifying peaks in the loss of the model can be used. The right block of Fig. 5.1 depicts this exact training procedure (*data adaptation phase*).

### 5.1.3 Evaluation

The capability of the network DT to replicate the physical system is measured by (i) generating similar data distributions and (ii) being able to react to distribution shifts.

#### 5.1.3.1 Digital Twin Performance

DT's performance is dissected into the performance of the decoding time distribution prediction task and the decoding success probability task.

**Decoding Time Prediction Task.** The left plot of Fig. 5.2 shows the normalized Probability Density Function (PDF) of the real distribution using the Kernel Density Estimation method as well as the predicted one for 15 dB of SNR, 40 PRBs, 100% CPU Capacity, and MCS index  $\in \{2, 12, 20\}$ . Looking at the real PDF, lower MCS indices imply lower decoding complexity, yielding smaller mean decoding times. Instead, higher MCS yields higher variability of the decoding times. This is attributed to the more complex task the decoder software implementation has to solve. This may incur glitches such as cache misses that reduce the overall performance unevenly, producing higher uncertainty. Performing the Kolmogorov-Smirnov test for various scenarios verifies the original modeling assumption to capture the decoding time with a normal distribution. For all the tested combinations of selected MCS, the null hypothesis was accepted with a 99% confidence interval.

**Decoding Success Probability Task.** Then, I evaluate the predictions of the decoding success probability task. The decoding probability is high in low MCS index and number of PRBs, since the data rate is low. When either the MCS or the number of PRBs increases, the carried data increases, which gets closer to the theoretical capacity

of the channel for this SNR level, and the probability gradually drops to 0. The BCE loss for 10 dB of SNR and a variety of combinations of MCS and PRB is shown in the right plot of Fig. 5.2. The DT manages to learn the trend when the probability is either very high or very low, while it gives a small prediction error in the transition region. Overall, the DT succeeded in representing the real LDPC decoder in both tasks, which facilitated the development of the ATHENA scheduler in various ways, such as evaluating and iterating faster between different RL architectures and algorithms.

### 5.1.3.2 Distribution Shift

In this experiment, I study the interaction between the PT and the DT, and the capability of the latter to adapt to previously unseen inputs. This may happen when the available historical data does not cover regions of the input space, *e.g.*, due to sudden failures in the CPU capacity or a new implementation or computing infrastructure that was not used to create the DT.

To simulate the former scenario, I draw the available CPU capacity of the PT in the interval 90-100% of the maximum achievable, *i.e.*, considering only very high CPU capacity (distribution  $\mathcal{A}$ ). The collected dataset is divided into the training set, used for training the DT, and the validation set, used for evaluating its performance across the training epochs. In Fig. 5.3, I plot the normalized validation loss when DT is trained for 30 epochs.

Then, the CPU is initially operating at the interval 10-90% of its maximum capacity (distribution  $\mathcal{B}$ ), *i.e.*, considering a wide range for the CPU capacity, in any case, lower than the previously used one. The retraining procedure can be triggered either at a certain frequency or when the similarity between the DT predictions and recent observations drops below a certain threshold. In this case, a distribution change  $\mathcal{A} \rightarrow \mathcal{B}$  took place, which caused a drop in similarity and issued a retraining of the DT. Note a spike in the validation loss as previously unseen observations give high prediction error, which later decreases as the model readjusts. Finally, I introduce a further distribution change by modifying again the CPU capacity to the full range of 10-100% (distribution  $\mathcal{C}$ ). Even though the new distribution has changed ( $\mathcal{B} \rightarrow \mathcal{C}$ ), it comprises inputs that have already been seen in the past and are already incorporated in the model, which explains the drop in the loss instead of a sudden increase.

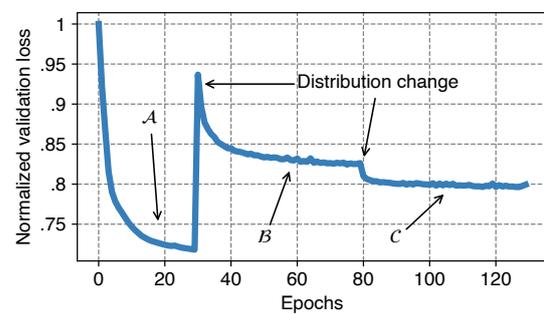


Figure 5.3: The DT validation loss (sum of BCE and NLL losses) when distribution changes happen.

## 5.2 Explainable AI

The second block of ATHENA is represented by the MR part, which interprets the results of the ML and devises (at a slower pace) actionable decisions on the network.

The model has to provide *insights* and *explanations* into its internal functionality and decision process. These explanations are categorized into three different classes depending on the question that they seek to answer [42]; the *attributive*, the *contrastive*, and the *actionable* explanations. In the study, I investigated methods for all three categories and adjusted them accordingly to the NN-based AC architecture.

### 5.2.1 Attributive Explanations

The attributive explanations answer to the question *why*, given an input, the model gave a certain outcome. These insights touch both the input, *e.g.*, features of a particular example, as well as the internals of the model (splitting rules in the case of a decision tree, neurons in the case of neural networks, *etc.*). They provide a comprehensive correlation between the input and the output, which does make sense for an external observer. This categorization mainly comprises feature importance methods such as LIME [95] and SHAP [96], which are model-agnostic explainers.

Model distillation is another method that can be used to produce explanations. The key concept behind it is to distill a *closed-box* model (such as a neural network) into an inherently better *explainable* surrogate model. This class of surrogate models typically comprises shallow decision trees and linear models, whose parameters (splitting criteria and variable coefficients respectively) are generally considered to be better understood. The distillation process consists of two models, namely the *teacher*  $\mathcal{T}$  and the *student*  $\mathcal{S}$ .  $\mathcal{T}$  is the closed-box trained model for which the explanations are asked.  $\mathcal{S}$  is a smaller surrogate model that integrates the knowledge of  $\mathcal{T}$ , while maintaining its accuracy.

ATHENA-ML’s model, which is described in Eq. 4.3, consists of two neural networks; the actor’s  $\mu_\theta$ , which provides an approximate solution  $\hat{\alpha}$ , and critic’s  $Q_\phi$ , which assists to refine it to  $\alpha$ . The actor’s model is distilled to a regression tree, which is a decision tree that holds linear models in its leaves (instead of constant approximators). Employing *response-based knowledge distillation* [97],  $\mathcal{S}$  mimics only the  $\mathcal{T}$ ’s output layer. During the training process,  $\mathcal{T}$  (actor) and  $\mathcal{S}$  produce outputs  $\hat{\alpha}_\mathcal{T}, \hat{\alpha}_\mathcal{S} \in \mathcal{R}^2$  respectively. To train the tree  $\mathcal{S}$ , these outputs pass through the interpolation function  $h : \mathcal{R}^2 \rightarrow \mathcal{R}$ . This interpolation function is created using TBS as the target value of combinations  $(n, m)$ . The outputs  $h(\hat{\alpha}_\mathcal{T}), h(\hat{\alpha}_\mathcal{S})$  are used to compute the *distillation loss*, which in this case is the  $L_2$ -distance, capturing the difference in yielded throughput between  $\mathcal{T}$  and  $\mathcal{S}$ . The teacher’s predictions serve as target values, and the decision tree  $\mathcal{S}$  is trained in a supervised way, using the distillation loss as split criteria. I used a maximum tree depth of 3, to make it easier to interpret.

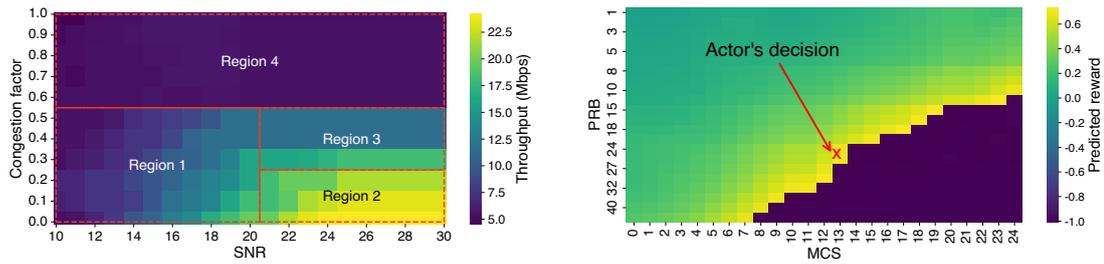


Figure 5.4: Left: ATHENA-ML throughput and the regions of the distilled decision tree. Right: Critic’s reward prediction for SNR 20 dB and congestion factor  $\beta = .6$ .

The left plot of Fig. 5.4 depicts the achieved throughput of the agent for combinations of the contextual features; the borders of the leaves of the trained decision tree are shown with the red dashed-edged rectangles. The interpretation of the splitting thresholds can facilitate the understanding of the internal decision process of the actor. The tree has divided the input space into 4 regions, in which the NN-based actor can be approximated by a linear model. These areas have a specific meaning in the context of a vRAN system. For congestion factors  $\beta > .5$  (Region 4), the actor yields the same throughput independently of the channel conditions. That is, the CPU is so congested that ATHENA-ML is forced to scale down consistently both the MCS and PRB for all channel conditions. For lower  $\beta$ , instead, the SNR is taken into account, and for values higher than 20 dB, the agent is divided into 2 different models, so that the maximum achievable throughput can be more finely approximated.

### 5.2.2 Contrastive Explanations

Contrastive explanations answer to the criticism *why not* a different result is the outcome of the AI system.

In ATHENA’s case, I look for the explanation for a specific decision taken by ATHENA-ML: *given a certain context  $c$ , why has  $\alpha = (n, m)$  been decided instead of  $\alpha' = (n', m')$ ?* The architecture and the objective function of the actor in Eq. 4.2, *i.e.*, to maximize the reward prediction of the critic, helps answer this question. Querying the critic  $Q_\phi(c, \alpha)$  and  $Q_\phi(c, \alpha')$  and comparing the reward expectation provides a quantitative explanation of the actor’s decisions. For example, in the right plot of Fig. 5.4, the contextual state  $\beta = .6$  and  $\bar{c} = 20$  dB, shows the critic’s reward prediction for all possible actions. From the figure, note that ATHENA-ML actor learns one of the actions that yield the highest predicted reward, effectively approximating in its internal variables where the negative reward area (*i.e.*, a frame not decoded in time) is hit.<sup>1</sup>

<sup>1</sup>It is important to clear out that the agent outputs directly the best action and does not infer the critic for all the possible actions, which would be intractable.

### 5.2.3 Actionable Explanations

Actionable explanations seek to answer *how* the input of the model should be modified so that a certain user-defined desired outcome is achieved. The change to be made is otherwise called *counterfactual*. This category of explanations is of particular importance from the network perspective because it allows not only understanding the system but also taking further decisions (*e.g.*, re-orchestration decision) to steer the system to a desired operational state, effectively implementing the vision depicted in Fig. 2.3. There may exist multiple counterfactuals that answer the question above, but the counterfactual that requires the smallest possible change [98] should be considered.

For this analysis, I focus on a specific counterfactual orchestration decision: *what* should be changed in the vRAN setting so that the throughput that ATHENA-ML achieves is at least  $thr_{target}$ , *e.g.*, a throughput Key Performance Indicator (KPI) in an eMBB scenario? From the network orchestration perspective, the only variable that the operator can control is the vRAN's CPU congestion factor  $\beta$ , since the user's channel conditions depend on many exogenous factors related to the UE characteristics. Because configuring the minimum  $\beta$  is the obvious answer, I introduce a sustainability clause in the cost function  $C(\beta) : \mathcal{R} \rightarrow \mathcal{R}$  of operating at a certain  $\beta$ , measured in monetary units, that forces the system to operate the system using the cheapest solution in terms of computing capability. Although the specific shape of this cost function can be very complex, there is only a simple requirement constraining  $C(\beta)$  to be a monotonically decreasing function. This is motivated by real systems, as high  $\beta$  implies the usage of a less expensive CPU (in terms of executed instructions per second) or a higher concentration of competing jobs on the same CPU set as the one the vRAN is running on, in a CPU sharing fashion. In this context, the counterfactuals are produced as a solution to the following optimization problem [98]:

$$\begin{aligned} \min_{\beta \in \mathcal{B}} \quad & C(\beta) \\ \text{s.t.} \quad & thr_{\text{ATHENA-ML}}^{\beta} \geq thr_{target} \end{aligned} \tag{5.1}$$

where the constraint assures that the throughput achieved by ATHENA-ML  $thr_{\text{ATHENA-ML}}^{\beta}$  at congestion factor  $\beta$  satisfies the target.

The MR block of ATHENA, namely ATHENA-MR, analyzes the different input states, over a time window, to take further decisions that target the optimization of *e.g.*, the computing resources. Consequently, the MR part runs in the Non-RT RIC, optimizing the orchestration by *e.g.*, (i) orchestrating the gNB decoder threads to a more congested CPU to save operational costs, or (ii) taking the opposite decision when they are running in a too congested infrastructure and the performance is not acceptable anymore.

A simple heuristic can solve this optimization problem. A replica of ATHENA-ML agent exists in the Non-RT RIC. ATHENA-MR acts reactively and considers the user's minimum SNR over the last window, *i.e.*, the worst-seen channel condition, iterates over the possible

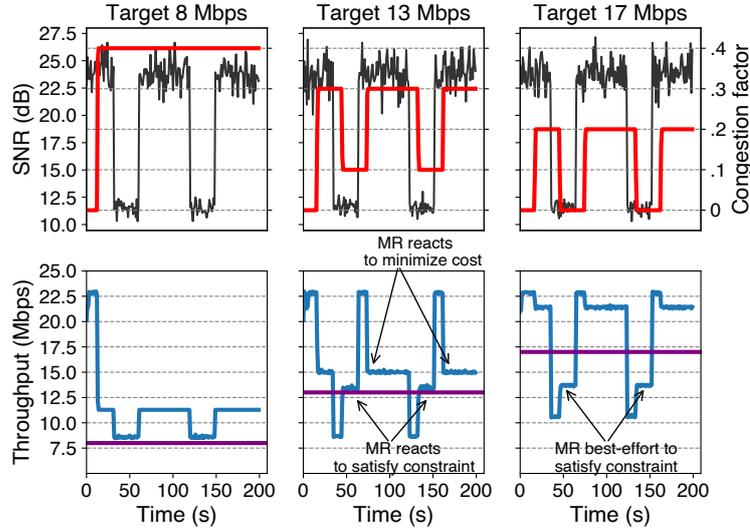


Figure 5.5: Congestion factor re-orchestration by ATHENA-MR to satisfy KPI.

$\beta_i \in \mathcal{B}$ , infers ATHENA-ML’s achievable throughput, and retrieves the minimum  $\beta$  that satisfies the constraint in Eq. 5.1.

To showcase the capability of the ATHENA-MR to re-orchestrate the congestion factor, consider a scenario of a UE at full buffer demand, where the gain  $G$  between the channel and the UE is modified periodically between .5 and .12 for 200 s. The time window over which MR operates is 30 s and initially  $\beta = 0$ , *i.e.*, operating at maximum operational cost. To mimic a network orchestrator that has to decide among a finite set of possible actions (*e.g.*, assigning a given number of CPU cores, each of them with a specific capacity), the feasible set of possible actions equals to  $\mathcal{B} = \{0, .1, .2, \dots, 1\}$ .

Expanding or restricting the feasible set affects the running time of ATHENA-MR, which is linear to the number of elements, and its running time is trivial to the time window. I consider 3 different scenarios with 3 desirable throughputs: 8, 13, and 17 Mbps. The top row of Fig. 5.5 shows in black the SNR per PRB and in red the re-orchestrated  $\beta$ . The second row plots the achieved throughput and the target throughput with a straight line.

In all three scenarios, the ATHENA-MR reorchestrates the computational resources to save operational costs and satisfy the constraints. Observe that in the case of 8 Mbps target throughput, the congestion factor is only once modified since the target can be reached at both SNR levels. At a target of 13 Mbps,  $\beta$  is accordingly adjusted, but throughput violations occur when the SNR drops since the orchestration algorithm has not yet reacted to the contextual changes. Finally, in the last scenario of 17 Mbps, the minimum throughput can not be even reached because of the low SNR; however, the MR reacts best-effort and lowers the congestion factor to 0, before recovering back to .2 when the SNR increases back again.

Fig. 4.2 shows how a closed-loop using MR is shaped. The O-DU informs *ATHENA-MR* about the worst seen SNR over the last time window via the O1 interface, which solves the optimization problem based on *ATHENA-ML* model and service requirements and configures the computing resources in the O-Cloud via the O2 interface [7], which provides a standard interface for intercommunication between the SMO and the O-Cloud.

## Summary

This chapter presented two complementary aspects that support the design, training, and deployment of *ATHENA*. First, a network DT of the LDPC decoder in a vRAN was developed. The DT, trained on extensive real-trace measurements, accurately reproduces decoding times and success probabilities under varying channel and CPU conditions. This enables fast and repeatable offline evaluation of different RL architectures for *ATHENA*, reducing the model development and evaluation lifecycle. Retraining mechanisms allow the digital twin to adapt to previously unseen input distributions, ensuring continued fidelity to the physical system.

Second, three explainable AI techniques were integrated to improve the interpretability of *ATHENA*'s scheduling decisions: attributive, contrastive, and actionable explanations. Using actionable explanations, which enable network-level interventions, a basic algorithm was developed to re-orchestrate CPU resources in the O-Cloud to meet throughput targets while minimizing operational costs.

# 6

## Qu4Fec: LDPC Decoding on Quantum Platforms

---

### Disclaimer

This chapter is based on the following publications:

- **Nikolaos Apostolakis**, Marta Sierra-Obea, Marco Gramaglia, Jose A. Ayala-Romero, Andres Garcia-Saavedra, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. 2025. Quantum Computing in the RAN with Qu4Fec: Closing Gaps Towards Quantum-based FEC Processors. *Proc. ACM Meas. Anal. Comput. Syst.* 9, 2, Article 36 (June 2025), 25 pages. <https://doi.org/10.1145/3727128>
- **Nikolaos Apostolakis**, Marta Sierra-Obea, Marco Gramaglia, Jose A. Ayala-Romero, Andres Garcia-Saavedra, Marco Fiore, Albert Banchs, and Xavier Costa-Perez. 2025. Quantum Computing in the RAN with Qu4Fec: Closing Gaps Towards Quantum-based FEC Processors. In *Abstracts of the 2025 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '25)*. Association for Computing Machinery, New York, NY, USA, 64-66. doi: 10.1145/3726854.3727311

Minor changes in structure and notation have been made to improve readability and consistency within this thesis.

In the previous chapters, I introduced the cloud-native system design for 5G core and RAN and used it to benchmark a CPU-aware MAC scheduler. While those chapters focused on the realization and benchmarking of vRAN components, this chapter explores a forward-looking approach that investigates the role of quantum computing in enhancing physical-layer processing. In particular, I study the feasibility of quantum annealing for LDPC decoding, a computationally intensive function within the uplink chain. Although the framework presented here is not integrated into the earlier vRAN testbed, this

chapter acts orthogonal and assesses the potential of quantum-based co-processors to be incorporated into vRAN systems, complementing traditional computing systems in future wireless networks.

This chapter introduces the design of an LDPC decoding solution suitable for a Quantum Annealer, named **Qu4Fec**. Developing this framework requires finding solutions for two aspects: *i*) a code design and *ii*) a QUBO formulation of the LDPC decoding problem. Both parts and design decisions are discussed in Sec. 6.1. Sec. 6.2 demonstrates the improvements achieved by the proposed solution and shows how its comparable performance to BP (introduced in Sec. 2.1.1.2) can make Quantum Computers an appealing option for next generation mobile networks, mostly due to their lowest energy footprint compared to traditional servers (introduced in Sec. 2.2.2). In the same section, I also identify clear limitations of the current and upcoming Quantum architectures in supporting LDPC decoding operations. Ultimately, in Sec. 6.3, I advance the body of knowledge about Quantum-powered FEC and point in several directions for the future development of practical Quantum RAN accelerators.

**Caveat on LDPC and Quantum.** Before proceeding further, I define the scope of this chapter, positioning it with respect to two completely different settings where coding and Quantum come together in the current scientific literature.

On the one hand, retrieving Quantum information has been challenging due to its noisy nature compared to classical bits. The noisiness of the Quantum platforms is ascribed to various factors: *i*) qubits are still not perfectly isolated from the environment and minimal interaction with the platform causes *decoherence*, degrading the Quantum state; *ii*) external disturbances can invert a qubit from  $|0\rangle$  to  $|1\rangle$  and vice versa; *iii*) additionally, in systems where qubits are entangled, an error in one qubit will propagate to others due to their strong correlation, exacerbating that effect [99]. In this context, significant efforts have been put into developing error correction codes that increase reliability when interacting with qubits [100]. These investigations can be intended as exploring FEC *for* Quantum and also use LDPC as an error-correction approach. Quantum LDPC codes are essential for building fault-tolerant Quantum systems, needed to scale up Quantum computers since they provide a high rate of error detection and correction. This study does *not* belong to this class of works.

On the other hand, Quantum technologies have been recently proposed as a processing tool to solve FEC problems [16] already present in the telecommunication area. That is, the idea is to leverage Quantum systems as an alternative to traditional computing solutions to tackle LDPC decoding, which is ubiquitous in the latest WiFi [101] and 5G NR standards [22]. Thus, these works solve LDPC *with* Quantum. This study and all the contributions listed above fall in this category.

## 6.1 Decoding in the Quantum Environment

QBP [16] is the baseline, which is the state-of-the-art Quantum-friendly LDPC decoder. In this section, I discuss the QUBO formulation (Sec. 6.1.1) and the code design part of the problem (Sec. 6.1.2), where I point out the detrimental effect that custom-designed codes that are primarily aimed at fitting the Quantum hardware, such as those utilized by QBP, have on the error-correcting capabilities. In contrast, I propose to use well-established code design methods. In Sec. 6.1.3 and Sec. 6.1.4, the alternative QUBO expression, employed by Qu4Fec, is formulated, which formally verifies the fundamental LDPC decoding problem in Eq. 2.1, which ultimately yields stronger guarantees and higher performance than that introduced by QBP.

### 6.1.1 LDPC QUBO Formulation

To use Quantum computing to solve the fundamental problem of LDPC decoding in noisy communication systems, a challenge arises in that classical iterative solutions such as BP cannot be executed on QA platforms. QBP [16] was the first solution in the literature to propose a QUBO formulation for LDPC codes that is suitable for subsequent solution by the QA. In the formulation proposed by QBP, the variables used in the QUBO problem are split into two types: *i*) the code variables  $[q_0, q_1, \dots, q_{n-1}]$  that denote the output decision of the decoder, and *ii*) the ancillary variables that describe the modulo-2 constraints enforced by the LDPC as an optimization objective.

The QUBO formulation for the LDPC decoding comprises two terms: *i*) the LDPC constraint term that weights the possible solution, penalizing not valid (*i.e.*, with no valid check constraints) codewords, denoted by  $Q_{LDPC}$ , and *ii*) a correlation function that associates the problem solution to the received channel values, denoted by  $Q_C$ . This latter term steers the QA solver toward solutions that are more related to the initial set of received values after the demodulation.

The final QUBO formulation implemented by QBP is a weighted summation of the LDPC constraint function and the correlation function

$$Q = a \cdot Q_{LDPC} + Q_C \quad (6.1)$$

where  $a$  is a positive weight factor, which steers the balance between the two terms. QA can find the solution that minimizes both terms, yielding a correctly decoded codeword.

Considering an  $(m, n)$  parity check matrix  $H$ :

$$Q_{LDPC} = \sum_{j=1}^m Q_{LDPC_j}$$

with  $Q_{LDPC_j} = \sum_{i=0}^{n-1} (H_{ji} \cdot q_i - 2 \cdot L_j)^2$  being the minimization function objective related

to the  $j$ -th constraint and  $L_j$  is a function of the ancillary qubits, which depends on the degree (number of 1's) of that constraint. The utility of  $L_j$  is to enforce that the modulo-2 (*i.e.*, XOR in binary) summation of the  $j$ -th constraint's constituent variable nodes sums to 0 and is expressed by

$$L_j = \sum_{s=1}^t (2^{s-1} \cdot q_e^{j,s})$$

where  $t$  is a function of the check node degree and  $q_{e_{j,s}}$  is the  $s$ -th ancillary qubit of the  $j$ -th constraint.

The second term of the QUBO is the correlation function. This term incorporates the channel output values and aims to associate them with the problem solution. Let  $y = [y_0, y_1, \dots, y_{n-1}]$  be the received channel vector. The authors in [16], set  $Q_C$  to minimize the distance

$$Q_C = \sum_{i=0}^{n-1} (q_i - Pr(q_i = 1|y_i))^2$$

Finally, in [16]  $a$  is set based on the transmission SNR, performing a hyperparameter search.

## 6.1.2 Code Design

### 6.1.2.1 Cycle Length

The error-correcting capability of a code is strongly correlated with the cycle length distribution of the Tanner graph, as the bit corrections rely on the independent message passing between the graph nodes [102,103]. Shorter cycles break this independence earlier in the decoding process, leading to decreased performance. Since an LDPC code always contains cycles in practical scenarios with finite code size  $n$ , the challenge is designing the parity check matrix so that the corresponding Tanner graph has cycle lengths as large as possible. Several efficient regular LDPC code design methods are available, such as Gallager [30] and PEG [104].

In the top part of Fig. 6.1, I illustrate the Tanner graph extraction from a parity check matrix, where  $H_{ij} = 1$  denotes the existence of a link between check node  $i$  and variable node  $j$ . In the bottom part of Fig. 6.1, I display the  $LLR_1$  update across one BP iteration. The message originating from  $v_1$  traverses 6 nodes before concluding at  $v_1$ , yielding a cycle length of 6. For BP to function optimally, the messages need to be independent. However, in the presence of unavoidable cycles in practical codes of finite length, this property is invalidated, and the performance of BP deteriorates, as explained next.

### 6.1.2.2 Strategies

FEC codes are carefully conceived to improve error-correcting capabilities in digital communications, and the error rate performance is the main metric that needs to be

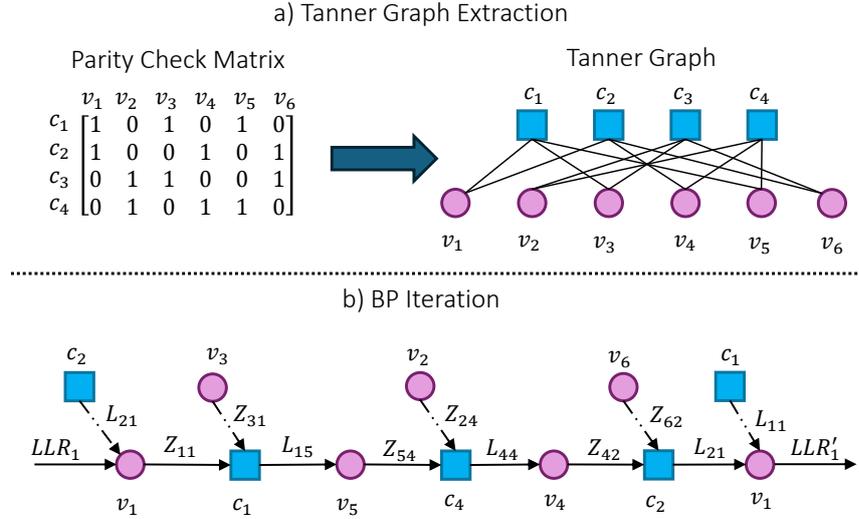


Figure 6.1: a) Tanner graph extraction from a parity check matrix. b) A BP update for  $LLR_1$ . The cycle length is 6 since the update traverses 6 nodes before returning to  $v_1$  ( $v_1 \rightarrow c_1 \rightarrow v_5 \rightarrow c_4 \rightarrow v_4 \rightarrow c_2 \rightarrow v_1$ ).

optimized in the development of an FEC code. Yet, in a QA, FEC code design needs to also consider the specificity of the execution platform (*i.e.*, the QPU) that will be used. Fig. 6.2 illustrates the natural order for this process: from (a) designing the LDPC code to (b) the extraction of the QUBO formulation (which produces the source graph) till (c) the embedding on the QPU target architecture (such as commercial QPUs like Chimera, Pegasus or Zephyr manufactured by D-Wave<sup>1</sup>) used for decoding.

The flow from (a) to (c) is not straightforward since the target graphs of cutting-edge QA platforms impose severe constraints that reflect on the QUBO and parity check matrix. The problem is so binding that QBP devises an LDPC code specifically tailored to D-Wave's Chimera QA framework by reversing the logical process: the design starts from the constraints in (c) and goes backward to a code design in (a). This turnaround has, however, the effect that the choices of QUBO formulations and LDPC design are critically curbed.

This point can be provided with a practical example. Using the QBP embedding method [16], I reconstructed the parity check matrix of a (2,3,420) code. For comparison, I also constructed a code with the same properties using the Gallager [30] method. For each of the codes, the generated codewords passed through an AWGN channel and were decoded using BP. The performance is measured in terms of BER/BLER, which is a fair comparison since the two codes add the same number of redundancy bits to the wireless channel transmission.

The comparison of both BER and BLER, depicted in Fig. 6.3, shows a noticeable

<sup>1</sup><https://www.dwavesys.com/>

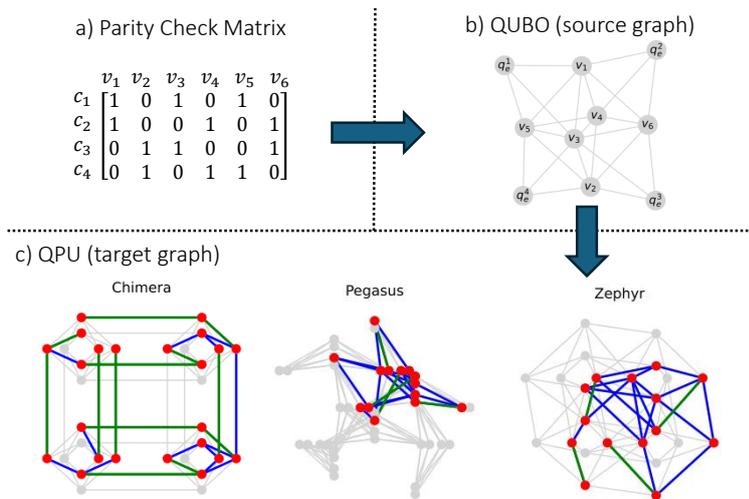


Figure 6.2: Steps from designing an LDPC code for Quantum decoding. *a)* Design the parity check matrix. *b)* Extract the QUBO formulation (source graph). *c)* Embed it into the QPU architecture (target graph). Red, blue, and green show the occupied qubits, the edges of the source QUBO, and the edges of the formatted chains, respectively. Qu4Fec uses the Gallager method to generate regular codes (step *a*) and D-Wave’s MM to embed to QPU (*b*  $\rightarrow$  *c*).

gap between the two strategies, with the Gallager code design outperforming that of QBP by almost an order of magnitude. Although the manually designed code in QBP benefits the target QA by occupying qubits as efficiently as possible, it disrupts the cycle length property, discussed above, and results in inoperable performance.

The effect is revealed in the right plot of Fig. 6.3, which measures the minimum cycle length of each variable node in the Tanner graph and plots the Cumulative Distribution Function (CDF) for the two cases considered, *i.e.*, QBP and Gallager. The median cycle length of QBP is half that of Gallager. As shorter cycles introduce higher correlations between the bits in the sequence to be decoded and limit the inherent capabilities of the code itself, the result highlights how the manual design of QBP inherently and substantially curbs the final decoding performance.

In light of these observations, I next propose a novel design for FEC codes in Quantum settings that primarily aims at preserving decoding capabilities. To this end, I abide by a traditional, sensible pipeline that starts from efficient codes created with the Gallager method.

### 6.1.3 Qu4Fec: Alternative QUBO Formulation

As mentioned in Sec. 6.1.1, the QUBO formulation proposed by QBP [16] requires the factor  $a$  to be computed and optimized based on prior hyperparameter tuning. This makes QBP SNR-dependent, relying on accurate SNR estimations, which are challenging to derive

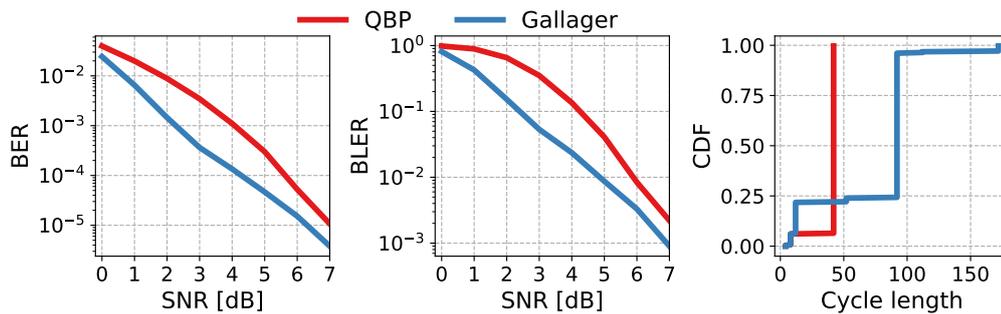


Figure 6.3: Comparison of QBP [16] and *Gallager* design method of (2,3,420) LDPC code. a) BER, b) BLER, c) Cycle length distribution CDF.

in operational settings [105]. Even then, however, it is not formally guaranteed that the minimum energy solution of the QA coincides with the maximum likelihood solution of the fundamental decoding problem. The proposed solution **Qu4Fec** is a formulation that provides formal proof without the need for prior hyperparameter optimization, as it works by only taking the received channel values as input.

The novel QUBO formulation of **Qu4Fec** derives directly from the ML objective detailed in Eq. 2.1 and avoids the hyperparameter tuning step. Considering BPSK modulation  $y_i = 1 - 2 \cdot x_i$  (where  $x_i$  is the bit to transmit,  $y_i$  is the transmitted symbol) over an AWGN memoryless channel. By expanding Eq. 2.1:

$$\begin{aligned} \hat{x} &= \arg \max_{x \in \mathcal{C}} \prod_{k=0}^{n-1} P(y_k | x_k). \\ \hat{x} &= \arg \max_{x \in \mathcal{C}} \sum_{k=0}^{n-1} \ln f_{Y|X}(y_k | x_k), \\ \hat{x} &= \arg \max_{x \in \mathcal{C}} \sum_{k=0}^{n-1} y_k (1 - 2x_k). \end{aligned} \quad (6.2)$$

Based on this, a novel QUBO formulation is derived for the LDPC decoding problem  $P_1$ . Using the same structure as Eq. 6.1 and redefining:

$$P_1 = a' \cdot Q_{LDPC} + P_C \quad (6.3)$$

by re-designing the correlation function as the minimization function of the negative ML function of Eq. 6.2 as

$$P_C = \sum_{k=0}^{n-1} -y_k (1 - 2 \cdot q_k)$$

Similar derivations can be made for higher-order modulations. For QPSK modulation following the 5G NR mapping [22], input bits  $x_{2k}, x_{2k+1}$  will be mapped to symbol  $s_k =$

$(1 - 2x_{2k}) + j(1 - 2x_{2k+1})$  where  $k = 0.. \frac{n}{2} - 1$ . The Eq. 6.2 will now become for QPSK:

$$\hat{x} = \arg \max_{x \in \mathcal{C}} \sum_{k=0}^{\frac{n}{2}-1} \text{Re}(y_k)(1 - 2x_{2k}) + \text{Im}(y_k)(1 - x_{2k+1})$$

where  $\text{Re}(\cdot), \text{Im}(\cdot)$  denote the real and imaginary part of the complex  $y_k$ . For the rest of the analysis, and without loss of generality, I will proceed by considering Binary Phase Shift Keying (BPSK) modulation.

The design objective is to determine a sufficiently large  $a'$  that prioritizes constraint fulfillment ( $Q_{LDPC}$ ) over correlation function optimization ( $P_C$ ). A heavy weight on  $P_C$  and hence basing the decoding process largely on the channel values may be counterproductive in many conditions.

Thus, I follow a similar approach to [106] and consider the correlation function directed by the ML decoder instead of the Euclidean distance. Consider a valid codeword solution vector  $\hat{q}$  and an invalid solution  $\tilde{q}$ . It holds that  $P_C^{max} = \sum_{k=0}^{n-1} |y_k|$ ,  $P_C^{min} = \sum_{k=0}^{n-1} -|y_k|$ , and that for any solution vector  $q$ :  $P_C^{min} \leq P_C(q) \leq P_C^{max}$ . Thus,

$$\begin{aligned} a' \cdot Q_{LDPC}(\hat{q}) + P_C^{min} &\leq P_1(\hat{q}) \leq a' \cdot Q_{LDPC}(\hat{q}) + P_C^{max} \\ a' \cdot Q_{LDPC}(\tilde{q}) + P_C^{min} &\leq P_1(\tilde{q}) \leq a' \cdot Q_{LDPC}(\tilde{q}) + P_C^{max}. \end{aligned}$$

Since  $\hat{q}$  is a valid codeword:  $Q_{LDPC}(\hat{q}) = 0$  and  $\tilde{q}$  is invalid:  $Q_{LDPC}(\tilde{q}) \geq 1$ . Also, the maximum energy of any valid codeword has to be lower than the minimum of any invalid one. Thus:

$$P_C^{max} \leq a' + P_C^{min} \Rightarrow a'_{min} = P_C^{max} - P_C^{min} = \sum_{k=0}^{n-1} 2 \cdot |y_k| \quad (6.4)$$

These derivations yield that the ML solution  $\hat{q}_{ML}$ , *i.e.*, the valid codeword that minimizes  $P_C$ , is also the one that minimizes  $P_1$  since

$$P_1(\hat{q}_{ML}) = P_C(\hat{q}_{ML}) \leq P_C(\hat{q}) = P_1(\hat{q}).$$

Unlike the  $Q$  formulation used in QBP [16], the proposed  $P_1$  formulation ensures that the solution that yields minimum energy is the ML codeword and removes the need for an offline optimization of  $a$ . To showcase that with an example, a (2, 3, 420) LDPC code is considered and a channel of SNR 2 dB. Given the transmitted maximum-likelihood (ML) codeword (CW), I generate the 256 solution space by freezing the ML CW's first 412 bits and considering any binary combination of the last 8 bits. For each of these generated possible solutions, I evaluate the energy level according to the  $Q$  (QBP) and  $P_1$  (Qu4Fec) QUBO formulations and I compute the minimum energy solution. Fig. 6.4 illustrates the energy level for the two formulations across the solution space. Notice that QBP's

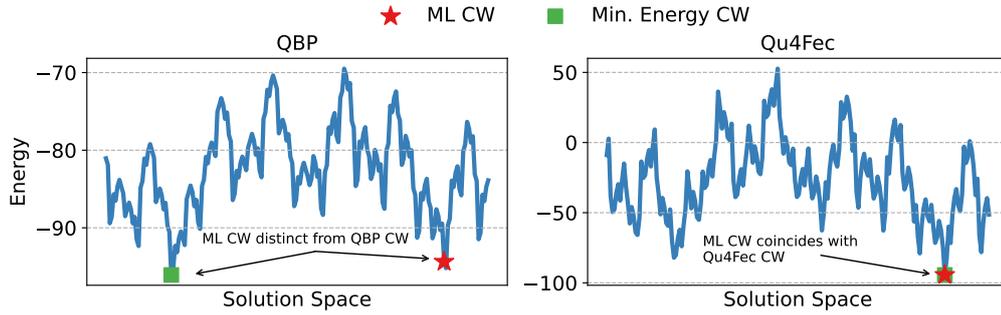


Figure 6.4: Sample comparison of QBP [16] and Qu4Fec energy levels. QBP formulation does not guarantee that the QUBO’s minimum energy codeword (CW) coincides with the Maximum Likelihood (ML) codeword. Qu4Fec formulation always reassures that its minimum energy codeword is the ML codeword.

minimum energy CW may be distinct from the ML CW, which will lead to a higher BLER, unlike Qu4Fec, whose formulation guarantees this property.

#### 6.1.4 Stability in Qu4Fec

Any codeword error increases the output QUBO energy by at least  $a'$ , which also depends on  $n$ . This creates a big energy gap between invalid and valid codewords, leading to instability in the annealing process, where the probability of avoiding local optima depends on the energy differential between the previous and the current state.

Consequently, the annealing process interprets any decrease in the energy as progress towards the global solution, with a very low probability of escaping these suboptimal states due to the energy decrease caused by  $a'$ . This issue inspired the derivation of a new formulation  $P_2$ , which restricts the energy differential between invalid, valid, and ML codewords.

However, each row in the parity check matrix functions as a parity check code itself (with just one row). Applying Eq. 6.3 for the  $j$ -th constraint

$$P_1^j = a'_j \cdot Q_{LDPC_j} + P_C^j,$$

with  $P_C^j = \sum_{k=0}^{n-1} -H_{jk} \cdot y_k \cdot (1 - 2 \cdot q_k)$  being the correlation function considering only the variable nodes of that constraint and  $a'_j$  weighting individually each LDPC constraint  $Q_{LDPC_j}$ . Following the same derivations as in Eq. 6.4:

$$a_j^{min} = \sum_{k=0}^{n-1} 2 \cdot H_{jk} \cdot |y_k|.$$

Since all constraints need to be optimized collectively, the Qu4Fec QUBO formulation

becomes

$$P_2 = \sum_{j=1}^m P_1^j. \quad (6.5)$$

$P_2$  achieves better numerical stability. For a single error in the achieved codeword at constraint  $j$ , the energy differential is  $a'_j$ , which considers the channel values only of that constraint. The previous formulation,  $P_1$ , used  $a'$ , which considered the whole  $n$  values. From the experiments,  $P_2$  led to better stability and performance closer to BP, as demonstrated in the next section.

## 6.2 Simulated and Experimental Evaluations

I study the practical performance of Qu4Fec, using QBP as a reference where appropriate, in two different settings, *i.e.*, via simulated annealing and with a real-world Quantum computer, as follows.

- First, in Sec. 6.2.1, I reproduce the Quantum process via Simulated Annealing [45], a stochastic optimization technique for approaching the global maximum of a QUBO. SA initializes a random solution to the problem, and in every step, picks a state close to the previous one. The system starts at a high-temperature state, where the probability of accepting a worse solution (a solution that increases the global energy) is high, facilitating the exploration of the energy landscape.

As the annealing process proceeds, the temperature cools down, this probability decreases, and the process slowly approaches the global minimum. Commercial QA platforms, such as the ones used in this paper, implement this algorithm using a physical system. Thus, SA can be regarded as the ideal annealer, and the success criterion for QA is how closely it can approximate the performance achieved by SA. Using SA as a first validation step disregards any physical limitations and implementation imperfections of the underlying quantum computing platform, focusing solely on the interactions between the qubits and couplers. Since the annealing process is inherently stochastic, both SA and QA perform a certain number of anneals for each submitted QUBO. Each anneal generates a solution, and the objective function is then evaluated for each solution to determine the achieved *energy*.

- Then, in Sec. 6.2.2–6.2.4, I implement Qu4Fec into a real-world QPU. The experiments are using the D-Wave Advantage [107] 6.1 platform on top of a hardware QPU with the Pegasus layout portrayed in Fig. 6.2. I extensively test its performance with a range of LDPC workloads and QA parameters (*e.g.*, annealing time, number of samples per anneal, chain strength [108], among others).

The results show, in fact, a significant discrepancy compared to the promising performance presented in Sec. 6.2.1 under SA. In particular, BER and BLER surged to almost 100% for code lengths greater than 60 bits, regardless of the SNR level.

Motivated by these QPU results, I investigate the root causes of the decoding performance drop, exploring the effect of the embedding algorithm and that of scaling and quantization. The analysis lets us explain some crucial steps that need to be taken before Quantum annealing takes place, the restrictions they impose, and their effect on decoding quality.

### 6.2.1 Benchmarking Qu4Fec via Simulated Annealing

Based on  $P_2$  and  $Q$  formulations, I demonstrate the superiority of the proposed solution Qu4Fec over QBP. For SA, the number of samples per submitted QUBO is fixed to 100, and the one with minimum energy gets selected, as an ideal annealing-based decoder would do. For each SNR level, I generate 10,000 message blocks, encode, modulate, and pass them through an AWGN channel.

BER and BLER are computed directly from the input and output codewords. Note that this contrasts with the approach in [16], which uses an *a posteriori* model based on the samples' distribution gathered from several executions on the same instance to estimate these metrics and has a harder applicability to a real-world decoder.

#### 6.2.1.1 Joint Effect of Code Design and QUBO Formulation

As discussed in the section above, Qu4Fec has two main differences with QBP; *i*) it relies on standard code design techniques instead of tailored code design to fit the QA (employed by QBP), and *ii*) it provides a novel QUBO formulation that builds on top of the fundamental decoding problem and does not require further hyperparameter tuning.

Fig. 6.5 shows how Qu4Fec can achieve 10% (set by 3GPP as the channel reliability threshold for enhanced mobile broadband services [36]) and 1% BLER with 1.82 db and 1.54 dB lower SNR respectively and can attain a  $7.6\times$  BLER improvement at 4 dB, for a (2,3,420) LDPC code.

#### 6.2.1.2 Effect of QUBO Formulation

Next, I isolate and corroborate the effect of the Qu4Fec's innovative QUBO formulation alone. Using the Gallager method, I generate LDPC codes and compare the BER/BLER curves on SA. Unlike the previous section, Qu4Fec and QBP are now evaluated on the same LDPC code. Additionally, BP's performance is also evaluated, which is considered the ground truth for traditional computing decoders, to quantify its discrepancy from Qu4Fec. For BP, I limit it to a maximum of 10 iterations, as further iterations provide only negligible improvements in BLER. I generate 3 different (2, 3,  $n$ ) LDPC codes where

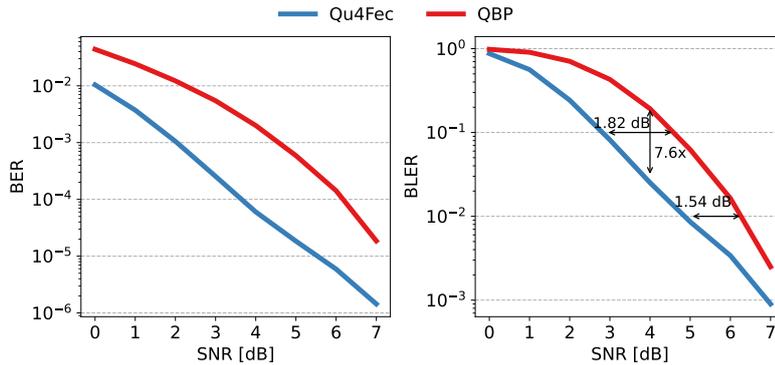


Figure 6.5: BER/BLER curves of Qu4Fec and QBP [16] for a (2,3,420) LDPC code.

$n \in \{105, 210, 420\}$ . Authors in [16] used 420 bits as the fixed codeword size, so I used a fraction as the baseline size in the benchmarks.

As illustrated in Fig. 6.6, the BLER discrepancy between BP and Qu4Fec is negligible across all code lengths. This demonstrates that the revised Qu4Fec QUBO formulation achieves comparable performance to BP<sup>2</sup>. However, as previously discussed in Sec. 2.2.2, this similar BLER performance can be achieved with a lower power footprint compared to BP, which operates on conventional computers. This makes Qu4Fec a promising alternative for FEC in the Quantum setting.

QBP performance diverges, especially in high SNR scenarios. This divergence arises from the fundamental approach of Qu4Fec QUBO formulation, where each invalid codeword yields higher energy than any valid one. QBP frequently fails in this regard, favoring invalid solutions that minimize the correlation function without giving the appropriate weight to the LDPC constraint satisfier component.

### 6.2.2 Effects of Non-ideal Embeddings

The embedding process is a crucial step of Quantum annealing on real platforms since it affects the total number of qubits participating in the process. It has been shown that a few dominant chains produced by MM, introduced in Sec. 2.2, hold significantly more qubits than the average chain in practical cases [109]. In the Noisy Intermediate-Scale Quantum (NISQ) era, each qubit is inherently noisy, so when more than one qubits are coupled together into a chain, the problem is exacerbated.

These imperfections in QAs arise from their analog nature, requiring operation at temperatures near absolute zero to enable Quantum phenomena like tunneling and

<sup>2</sup>I also evaluated the performance of commercial solvers, such as CPLEX, with varying parameter settings, in solving the QUBO. However, their performance was substantially inferior to that of SA. For instance, at 6 dB, CPLEX resulted in a 10% higher BLER. This is attributed to CPLEX's limitations in solving non-convex problems with a large number of variables, such as the LDPC decoding problem QUBO.

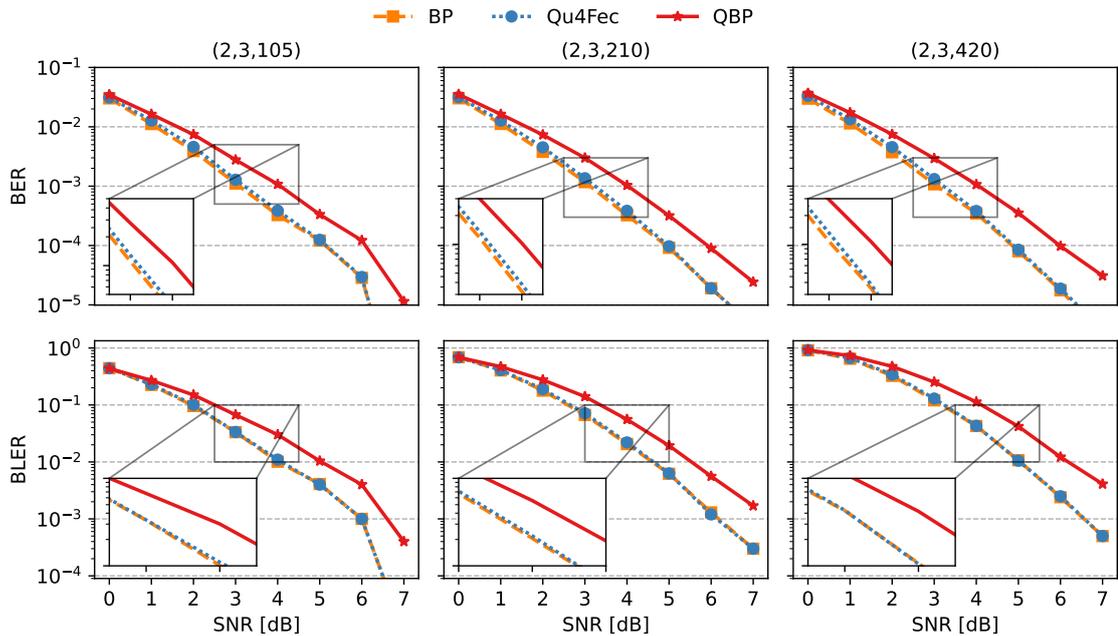


Figure 6.6: BER/BLER curves for BP, Qu4Fec, QBP and different code lengths.

entanglement. Controlling complex phenomena in extreme conditions is challenging and can lead to deviations from the intended representation of a given problem. These deviations are collectively referred to as Integrated Control Errors (ICE).

To better illustrate the issue, consider a problem with  $N$  binary variables, where  $h_i$  and  $J_{i,j}$  represent the linear and quadratic terms respectively, with  $i, j = 1, \dots, N$  and  $i < j$ . ICE can be modeled as errors  $\delta h_i$  and  $\delta J_{i,j}$  that affect the precision of these coefficients, thereby altering the original problem definition. Consequently, these errors can modify the objective function that the QPU is attempting to optimize.

To counter this behavior, the authors in [16] resorted to a manual embedding solution supported by the relatively easy (hence non-ideal) structure of the LDPC code they designed for QBP. The LDPC parity check matrix perfectly embeds on the D-Wave 2000Q QA, shipped with the Chimera architecture. This method allows QBP to fit larger codewords with fewer qubits due to the limited chain length.

As discussed in Sec. 6.1.2, this comes with a high price in terms of maximum cycle lengths and overall decoding performance. However, manual embedding faces more problems. The first is the generality of the solution: each generation of QA has a short lifetime, with frequent changes in the underlying architecture that are not backward-compatible and render previous manual embeddings inapplicable and their associated LDPC code design obsolete. Indeed, newer QPU platforms are not necessarily a superset of previous platforms: for example, D-Wave 2000Q is based on the Chimera  $C_{16}$  lattice, while D-Wave Advantage's Pegasus architecture is a supergraph of Chimera  $C_{15}$ . My

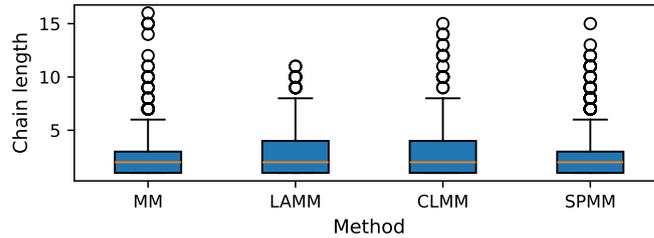


Figure 6.7: Qubits' chain length distribution using different embedding methods.

attempt to fit the  $(2,3,420)$  code of [16] onto the newest architecture using the QBP manual embedding was unsuccessful due to the bigger chain lengths that were created. Finally, embedding optimization for a specific code compromises generality, limiting its applicability to codes with different parameters.

Qu4Fec offloads the embedding task to the standardized and well-evolved MM, effectively decoupling the code design phase from the graph embedding. This yields many advantages, including versatility in accommodating diverse (and efficient, *e.g.*, Gallager) parity check matrices and invariance to the target graph's structural properties.

In the proposed Qu4Fec implementation, I used D-Wave's open-source MM [110]. Different variants of this algorithm were also investigated, such as LAMM [111], SPMM, and CLMM [112], which aim to optimize the mapping process by exploiting the structural characteristics of the source graph. Fig. 6.7 shows that the different embedding methodologies do not consistently improve the chain length distribution for a reference  $(2,3,420)$  code mapped onto the Pegasus layout. The reason is that the QUBO source graph of an LDPC decoding problem does not present a specific layout that those variants could leverage to render the embedding more efficient. Thus, for the rest of the chapter, MM is considered to be the embedding method.

### 6.2.3 Scaling and Quantization

The qubits and couplers in a QPU operate within specific ranges. For instance, the qubit bias range for the latest Advantage QPU is between  $-4$  and  $4$ , and the coupler strength ranges from  $-1$  to  $1$ . Since the coefficients of an objective function can theoretically span from negative to positive infinity, they must be scaled down to fit within these hardware constraints. Before programming the QPU, the coefficients are adjusted with a scaling factor to ensure they do not exceed the platform's acceptable range, leading to higher inaccuracies when coefficients have a larger span.

Following the programming of biases and couplers, a quantization process with finite resolution is applied. This process inherently introduces error, as closely valued terms might be quantized to the same value, potentially leading to significant distortions in the solution quality. In some instances, these distortions can dramatically impact the energy

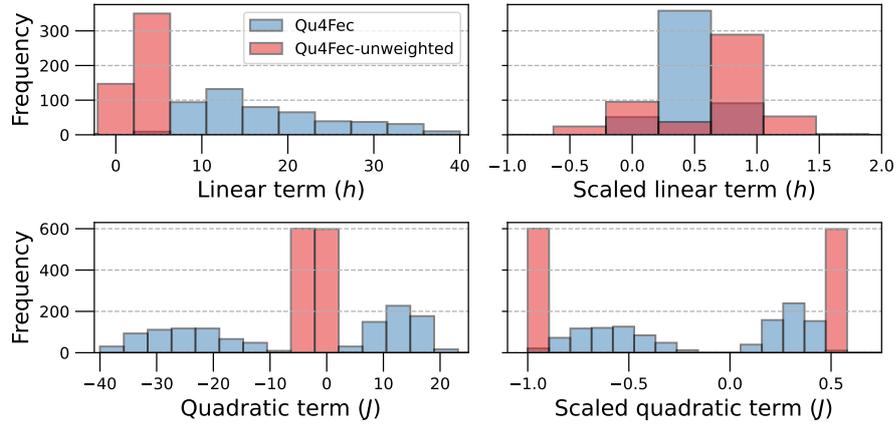


Figure 6.8: Original, scaled linear, and quadratic term histogram for the Qu4Fec and Qu4Fec-unweighted QUBO formulations.

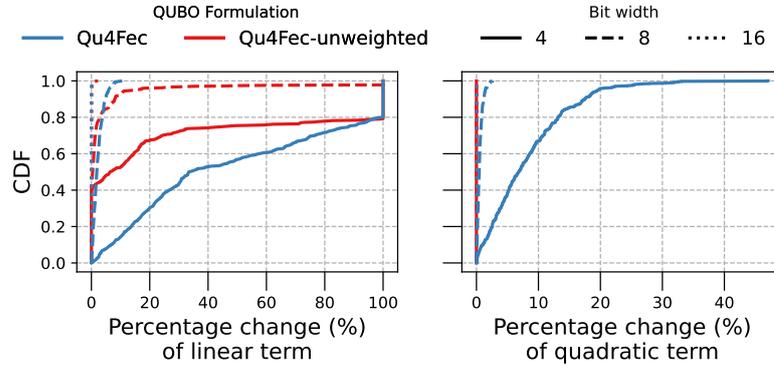


Figure 6.9: Percentage change of linear and quadratic terms due to scaling and quantization effect for Qu4Fec and Qu4Fec-unweighted formulations. Considered 3 different bit widths of 4, 8, and 16 bits.

landscape of the objective function, with even a slight change in value or sign altering the optimal solution.

While quantization errors occur independently of scaling, experiments indicate that scaling before quantization amplifies these errors. This suggests that when terms are first scaled, the subsequent quantization can lead to greater inconsistencies, further complicating the problem-solving process on a QA.

To demonstrate this effect, I introduce an unweighted version of the Qu4Fec QUBO formulation (Qu4Fec-unweighted) where  $a'_j = 1$ . Setting  $a'_j = 1$  effectively reduces the heterogeneity and range of the coefficients, simplifying the problems linked to scaling and quantization since they will now face similar modifications. However, this worsens the decoding performance as it does not optimize as per  $P_2$ .

Fig. 6.8 shows the histograms of the linear and quadratic coefficients for both QUBO formulations when injected into the Quantum platform within the above-mentioned range

$-4 \dots 4$  and  $-1 \dots 1$  for linear and quadratic terms, respectively. The variability of the unscaled linear and quadratic terms of the Qu4Fec formulation is disrupted when a scaling factor is applied, hence changing the factors of the problem.

The computation of the effect of the scaling process is impossible without access to the actual QPU hardware implementation to understand how the real-valued variables are quantized into the couplers. Since I do not have such access to the QPU, I considered three different bit resolutions (4, 8, and 16 bits). In Fig. 6.9, I show the CDF of the percentage change of both the linear and quadratic terms after quantization has taken place.

The quadratic terms (which exist only in the LDPC satisfier term) of the unweighted version incur a median error of 8% for 4-bit resolution and near-negligible error for higher resolutions, as these terms are not multiplied by the real numbers  $a'_j$ . However, notice a much larger discrepancy in the linear terms. For the lowest resolution, some terms may even suffer a 100% error, and only at very high resolutions is the accuracy high enough to avoid errors, indicating the high sensitivity of the linear terms to the scaling and quantization effect.

#### 6.2.4 Effects on the QA process

The errors introduced by embedding, scaling, and quantization affect the overall quality of the annealing process when executed on a physical platform. To quantify this effect, I compare Qu4Fec with its unweighted version on SA and QA and compute the discrepancy between the two processes in the left plot of Fig. 6.10. The relative energy discrepancy is computed as the energy difference between the solutions found by QA and SA divided by the ones found by SA. Observe that an overall easier problem such as Qu4Fec-unweighted always has a very low discrepancy between the lowest energy solutions, indicating how the inaccuracies introduced by quantization on the QA platform have a minor impact. As SA is considered the *ideal* annealer, higher discrepancies such as the ones shown for larger codeword sizes are likely due to the combined effect of quantization, scaling, and embedding.

Especially the latter has a noticeable effect, which is captured in the right plot of Fig. 6.10, where I measure the maximum chain lengths and number of occupied qubits experienced by the QPU for various codelengths. When the chain lengths are limited, even the more complex Qu4Fec still attains comparable results to SA when executed with QA.

Throughout this analysis, it is evident that current QA platforms still have room for improvement before they can give reliable results when employed as wireless processors due to the two error sources analyzed before: *i*) the embedding process, which affects the total number of qubits that will be involved in the annealing procedure, and *ii*) the resolution of the coefficients of the QUBO when they are programmed on the actual QPU.

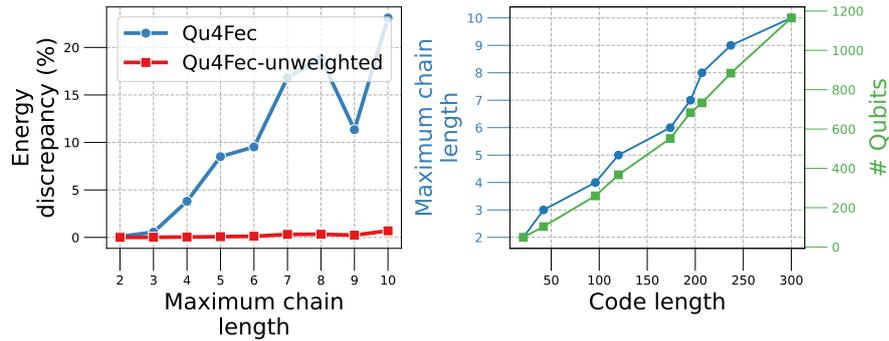


Figure 6.10: Left plot: Percentage energy discrepancy between QA and SA for Qu4Fec and Qu4Fec-unweighted QUBO formulations. Right plot: Maximum chain length and number of occupied qubits achieved by MM for various code lengths.

## 6.3 Towards Quantum-based Baseband Processors

By executing Qu4Fec on a commercial platform, I discussed how state-of-the-art QPUs are still far from being used as baseband processors, mostly due to the difficulties in the embedding process. In this section, different qubit fabric structures are explored to accommodate LDPC codes, discussing promising design trends for a Quantum wireless processor design.

### 6.3.1 Embedding Algorithms Analysis

Embedding algorithms such as MM take the LDPC source graph (*i.e.*, the one resulting from the code design) as input and map it onto the target one, resulting from the hardware design of the QPU. While the experimental evaluation used the Pegasus architecture, I now analyze the impact of the Chimera (past) and Zephyr (future) D-Wave architectures on the embedding process by measuring the resulting maximum chain length in Fig. 6.11. Subsequent QPU generations show that they *do* improve the embedding process. However, only very short codeword source graphs can be embedded with a maximum chain length that is sufficiently short to avoid energy discrepancy with Qu4Fec. According to the left plot of Fig. 6.10, the maximum chain length should not exceed a value of 3, outlining the gray region in Fig. 6.11, and codewords must be shorter than 40 bits to allow so even with the most recent Zephyr QPU architecture. More realistic [113] codeword lengths start to suffer a linear increase, with the newest generation only marginally (17% on average) improving the embedding process.

### 6.3.2 Optimal QPU Graph Structure

A tailored QPU architecture for LDPC decoding guarantees that no chains are employed at all during the embedding. This requires understanding the characteristics of

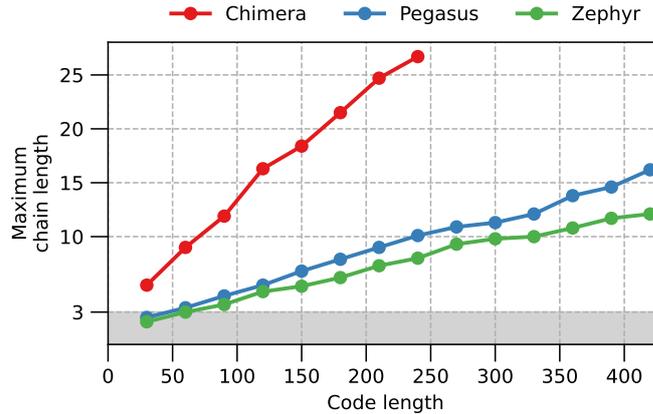


Figure 6.11: Maximum chain length achieved by MM for different code word lengths and target QPU architecture.

the source graph. To embed any LDPC code with a source graph of  $n$  total nodes and  $l$  ancillary nodes, a full mesh of  $n + l$  nodes is required. Even a full mesh of just  $n$  nodes is already impractical. For instance, to perfectly embed any  $(2,3)$ -regular codeword of up to length  $n$  as the ones that are studied, the target graph should have *i)*  $l = \frac{2n}{3}$  for ancillary variables, which are completely unconnected between them, and *ii)*  $n$  nodes for the code variables, forming a full mesh to accommodate any possible parity check matrix structure. These two sets of nodes are interconnected as a complete semi-bipartite graph with  $\frac{n(7n-3)}{6}$  edges. However, the empirical findings show that typically less than 1% of these edges are employed, indicating an extremely inefficient utilization.

### 6.3.3 Zephyr Layout Parameterization

QPU architectures are designed to facilitate QUBO problem embeddings while being practical and extensible. For instance, D-Wave’s next-generation Zephyr QPU follows this principle by creating a qubit fabric characterized by a concatenation of tiles so that QPU target graphs can be effectively represented by two parameters: the tile replication factor  $m$  and the internal tile pattern  $t$ . In a nutshell,  $m$  controls the size of the QPU, while  $t$  matches the available edges in a single tile. As a reference, in the former-generation Chimera architecture, the unit cell consisted of a bipartite graph of two shores of nodes, each of size  $t = 4$ . Pegasus and Zephyr retained this parameter but added sparse connections between nodes on the same shore, known as *odd couplers*.

To understand the capabilities of the Zephyr architecture, I computed the maximum chain length distribution achieved by MM under different values of  $m$  and  $t$ . The current Zephyr topology implementation from D-Wave offers  $t = 4$ , so extending this parameter allows to infer performance for soon-available QPUs.

By fixing  $t$  and varying  $m$  (as shown in the left plot of Fig. 6.12), it shows that if  $m$

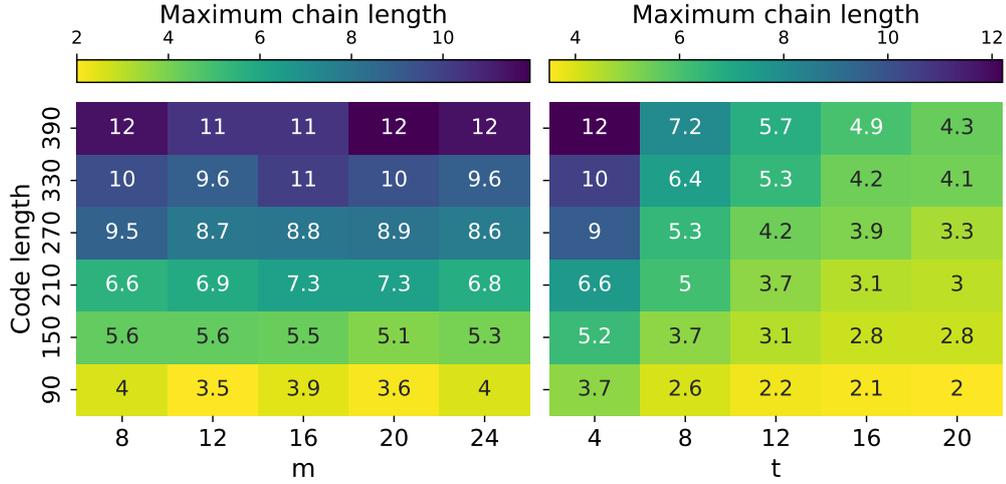


Figure 6.12: Left: Effect of  $m$  on the maximum chain length, with a fixed  $t = 4$ . Right: Effect of  $t$  on the maximum chain length with a fixed  $m = 8$ .

is large enough, then the advantages brought by extending the size of the architecture are null. This hints at the internal structure of the tile as the most important parameter. I further corroborate this in the right plot of Fig. 6.12, which shows how increasing  $t$  effectively reduces the maximum chain lengths as more edges between nodes are available. Still, the gap between current implementations and those that should offer dependable performance (*i.e.*,  $t \geq 20$ ) is large.

### 6.3.4 QPU Fabric Extensions

As not even the evolution of current commercial QPU platforms can reliably embed LDPC QUBO problems, I next investigate how they should be extended towards this goal. The key factor in decreasing the maximum chain length is the QPU's internal connectivity factor. Increasing connectivity within a tile (as shown in Fig. 6.12) offers a sublinear improvement on the maximum chain length (*i.e.*, from 12 to 4.3 by making the graph 5 times more complex), hence adding couplers *between tiles* is proposed.

In the commercial Zephyr, qubits in a tile connect to the corresponding qubits in the neighboring tiles using *external couplers*. To extend this design, I denote with a Zephyr- $k$  the Zephyr topology where each qubit in a tile connects to all the neighboring tiles up to  $k$ , as depicted in the top plot of Fig. 6.13.

The bottom plot of Fig. 6.13 shows the results of the embedding process for several codeword sizes and  $(m, t) = (8, 4)$ . Increasing the inter-tile connectivity impacts the maximum chain length and offers an easier target graph for embedding the Qu4Fec QUBO problem. A Zephyr-2 architecture would introduce 8% more edges than the original Zephyr, while the Zephyr-8, which may be effective for LDPC decoding, results in a constant 28% increase in the number of edges.

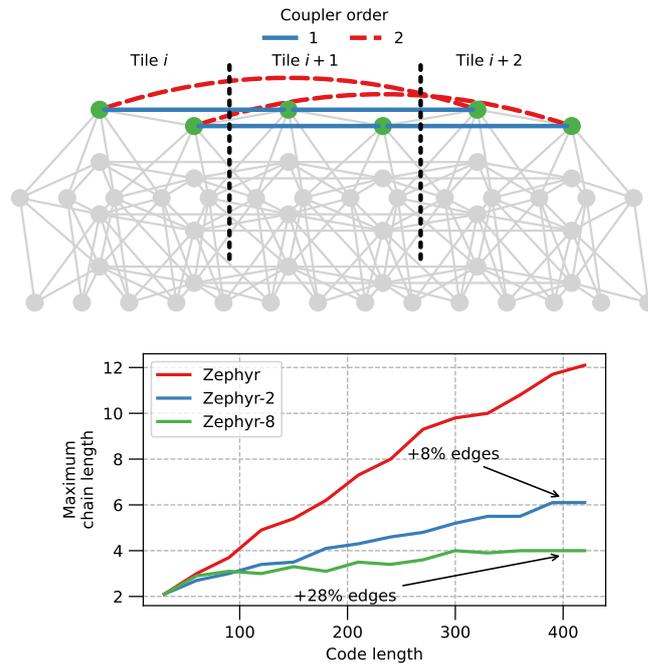


Figure 6.13: Top: Qubit in tile  $i$  connects to qubit in tile  $i + 1$  via first-order coupler (original Zephyr) and to qubit in tile  $i + 2$  via second-order coupler. Bottom: Effect of introducing higher-order couplers on the maximum chain length.

## Summary

In this chapter, I introduced **Qu4Fec**, a quantum computing approach for LDPC decoding. The contribution centered on two key components: *i*) the design of suitable codes and *ii*) the formulation of LDPC decoding as a QUBO problem. In simulated annealing experiments, **Qu4Fec** outperformed the state-of-the-art **QBP** by 1.82 dB and achieved performance comparable to the **BP** algorithm on classical systems. Considering recent evidence that quantum computers can operate at significantly lower power levels than conventional CPUs, **Qu4Fec** holds promise as a potential replacement for CPU- or ASIC-based LDPC decoders in future vRAN systems, potentially reducing both power and memory footprints.

When executed on current quantum annealers, however, **Qu4Fec** exhibited notable performance degradation due to the inherent noise and limited reliability of today's quantum hardware in encoding QUBO formulations. To address this challenge, I proposed extensions to the qubit layout that could better accommodate LDPC structures in future quantum platforms. These improvements aim to close the gap between simulated and hardware performance, paving the way for more practical integration of quantum computing in next-generation wireless systems.

# 7

## Conclusions and Perspectives

---

The heterogeneity of cloud resources mandates the development of new algorithms and paradigms that enable financially and energy-efficient resource utilization in next-generation mobile networks. On one hand, accelerators such as ASICs and FPGAs are difficult to orchestrate, manage, and share due to their rigid and specialized nature. On the other hand, recent research has proposed leveraging GPUs, which can be shared between 5G and AI workloads, to amortize both the initial investment and the considerable power consumption and operational costs of these high-throughput machines. However, this approach remains largely theoretical and has yet to be proven in practice. Therefore, there is a need to explore alternative paradigms that can further improve the resource efficiency of cloud systems supporting mobile networks.

This thesis addresses the above challenges through contributions in two main directions. First, it introduces a mechanism for reliably allocating the Virtualized DU (vDU) on Commercial Off-The-Shelf (COTS) servers, as an alternative to the ASIC-FPGA-GPU paradigm. Having proven their financial sustainability for long-term operations, COTS facilitates efficient resource sharing among multiple tenants. Second, it investigates the feasibility of executing specific components of the physical layer pipeline—specifically, Forward Error Correction (FEC) decoding—on quantum machines instead of traditional compute systems. By integrating quantum hardware into the cloud resource pool, the thesis explores a novel direction supported by studies suggesting that quantum machines may offer a sustainable path to meet the growing computational demands of 5G and future mobile networks.

### 7.1 Conclusions

This thesis explored the feasibility of deploying COTS CPU-based systems to support baseband processing in 5G and beyond mobile networks, offering a more manageable and shareable alternative among different tenants and applications.

In the first part of this thesis, a cloud-native 5G testbed was rigorously developed

and analyzed. At the time this work was carried out, it represented the first open-source deployment of a 5G core using Docker and Kubernetes in a fully cloud-native manner. By leveraging existing open-source projects and developing new modules and tools, I designed and implemented an end-to-end cloud-native 4G/5G mobile network. This platform provides a cost-effective environment for developing and validating novel control and orchestration algorithms, such as ATHENA, without requiring specialized hardware. The two use cases of dynamic network function orchestration and handover mobility demonstrated seamless user throughput, highlighting the platform's practical utility for researchers and practitioners seeking to experiment with flexible and dynamic network scenarios.

Building on this cloud-native testbed, Chapter 4 presents ATHENA, a machine learning framework for radio resource management in vRAN systems. ATHENA optimizes throughput while ensuring compliance with the baseband processing pipeline's timing requirements, according to the compute quota allocated by the underlying cloud infrastructure, and explicitly accounts for the *noisy neighbor* effect. The framework adheres to 3GPP and O-RAN standards, and by leveraging real-time CPU quota information, it achieves high reliability levels and maintains throughput even under high congestion scenarios, outperforming the baseline radio resource scheduler, whose throughput collapses to zero in the same conditions.

In Chapter 5, to support the development of ATHENA and enable rapid experimentation with different algorithms and architectures, I implemented a digital twin of its core component, the Low-Density Parity-Check (LDPC) decoder. The digital twin accurately mirrored the physical decoder's data distribution, as validated by Kolmogorov-Smirnov tests with a 99% confidence interval. Furthermore, to gain deeper insights into the internal functionality of ATHENA's machine learning components, this chapter also addresses the system's explainability, answering questions such as *why did ATHENA make a particular decision?* and *why did it not choose a different one?* In addition, I introduce the concept of *actionable* machine reasoning, wherein the system makes subsequent decisions based on the outcomes produced by ATHENA. On this front, a basic algorithm was developed to re-orchestrate CPU resources to meet throughput targets while minimizing operational expenses, demonstrating fast and context-aware responses to changing channel conditions.

Chapter 6 presents Qu4Fec, a solution for LDPC decoding using Quantum Processing Units (QPUs). The proposed LDPC decoder achieves a BLER improvement of 1.82 dB over state-of-the-art approaches and delivers similar performance in simulation to Belief Propagation (BP), a traditional computing algorithm for FEC. However, when implemented on an actual quantum computer, the decoder's error-correcting performance significantly deteriorated, indicating severe limitations imposed by the hardware platform. To investigate the root causes of this behavior, I analyzed the effects of scaling, quantization, and the embedding algorithm itself, all of which were found to adversely

impact wireless channel coding, rendering it currently impractical on commercial QPUs. In this context, I examined how enhancements to existing quantum platforms could improve performance and identified key internal connectivity characteristics that a QPU must possess to support FEC operations in future wireless infrastructure.

## 7.2 Future Perspectives

The work presented in this thesis lays the groundwork for rethinking how heterogeneous cloud resources, ranging from general-purpose CPUs to specialized quantum processors, can be leveraged to support sustainable and high-performance mobile network infrastructures. Building upon these findings, I hereby provide directions for future research.

Future work can expand the scope of the current cloud-native testbed to incorporate emerging classes of accelerators, such as FPGAs and ASICs. Although these platforms were initially excluded due to their orchestration complexity and cost, recent advances in virtualization and containerization technologies could enable more dynamic and programmable integration of such accelerators. For this purpose, the cloud-native architecture introduced in Chapter 3 provides a foundation that can be extended by researchers and practitioners to implement Kubernetes Custom Resource Definitions (CRDs) supporting additional accelerators.

Furthermore, while this thesis focused on ML-based radio resource management through ATHENA based only on the available CPU quota, future efforts could extend the framework to incorporate more fine-grained telemetry and workload prediction into the ML loop. Such metrics could be the nature of the third applications, *e.g.*, best-effort vs latency-sensitive, that would allow ATHENA to respond even more intelligently to changing contextual conditions. By being published as open-source, the ATHENA MAC scheduler can be extended to incorporate these metrics through modifications to the communication interface between the srsRAN scheduler and the ML module.

Regarding the FEC codes themselves and their evolution across the mobile network generations, the transition from 4G LTE to 5G NR saw Turbo codes replaced by LDPC codes, largely due to the high degree of data parallelism offered by parity-check matrices, which aligns well with the computational capabilities of modern CPU, GPU, and FPGA/ASIC architectures [114]. Protograph-based LDPC codes [115], which are already standardized in 5G/NR, are well-suited to the SIMD programming paradigms supported by the aforementioned platforms. This combination significantly enhances algorithm performance, resulting in higher throughput.

QPUs, however, exhibit heterogeneous computational characteristics that are fundamentally different from, and often orthogonal to, those of conventional computing platforms. In this context, two complementary research directions are emerging at the

intersection of FEC and quantum computing. First, as derived from Chapter 6, current quantum computers are unable to handle standard LDPC codes reliably. To fully harness the potential of QPUs, a novel family of codes could be designed specifically to exploit the unique optimization capabilities of quantum annealers, such as their ability to efficiently find global minima in high-dimensional spaces, while considering the constraints imposed by the qubit fabric layout. Conversely, and orthogonal to the first direction, extensions of the QPU fabric itself could be envisioned, similar to the proposals introduced in Chapter 6, targeting application-specific LDPC codes. Such hardware-aware adaptations could reduce the number of required qubits and, consequently, the noise introduced during computation, enabling more reliable quantum decoding operations.

Finally, as the computational landscape continues to evolve, a data-driven and adaptive orchestration layer will be essential to manage the increasing heterogeneity. The testbed introduced in this thesis can serve as a foundational layer upon which scalable and modular orchestration systems can be built—capable of dynamically allocating workloads across CPUs, GPUs, FPGAs, ASICs, and QPUs based on performance, energy, and latency trade-offs.

## References

---

- [1] N. Apostolakis, M. Gramaglia, and P. Serrano, “Design and Validation of an Open Source Cloud Native Mobile Network,” *IEEE Communications Magazine*, vol. 60, no. 11, pp. 66–72, 2022. [Online]. Available: <https://doi.org/10.1109/MCOM.003.2200195>
- [2] N. Apostolakis, M. Gramaglia, L. E. Chatzieftheriou, T. Subramanya, A. Banchs, and H. Sanneck, “ATHENA: Machine Learning and Reasoning for Radio Resources Scheduling in vRAN Systems,” *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 2, pp. 263–279, 2024. [Online]. Available: <https://doi.org/10.1109/JSAC.2023.3336155>
- [3] N. Apostolakis, L. E. Chatzieftheriou, D. Bega, M. Gramaglia, and A. Banchs, “Digital Twins for Next-Generation Mobile Networks: Applications and Solutions,” *IEEE Communications Magazine*, vol. 61, no. 11, pp. 80–86, 2023. [Online]. Available: <https://doi.org/10.1109/MCOM.001.2200854>
- [4] N. Apostolakis, M. Sierra-Obea, M. Gramaglia, J. A. Ayala-Romero, A. Garcia-Saavedra, M. Fiore, A. Banchs, and X. Costa-Perez, “Quantum Computing in the RAN with Qu4Fec: Closing Gaps Towards Quantum-based FEC Processors,” *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 9, no. 2, Jun. 2025. [Online]. Available: <https://doi.org/10.1145/3727128>
- [5] N. Apostolakis, M. Sierra-Obea, M. Gramaglia, J. A. Ayala-Romero, A. Garcia-Saavedra, M. Fiore, A. Banchs, and X. Costa-Perez, “Quantum Computing in the RAN with Qu4Fec: Closing Gaps Towards Quantum-based FEC Processors,” in *Abstracts of the 2025 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS/PERFORMANCE '24. Stony Brook, NY, USA: Association for Computing Machinery, 2025, pp. 59–60. [Online]. Available: <https://doi.org/10.1145/3726854.3727311>
- [6] Business Research Insights, “Virtualized Radio Access Network (vRAN) Market Size, Share, and Trends, 2024-2031,” 2024.

- [Online]. Available: <https://www.businessresearchinsights.com/market-reports/virtualized-radio-access-network-vran-market-106129>
- [7] A. Garcia-Saavedra and X. Costa-Perez, "O-RAN: Disrupting the Virtualized RAN Ecosystem," *IEEE Communications Standards Magazine*, vol. 5, no. 4, pp. 96–103, 2021. [Online]. Available: <https://doi.org/10.1109/MCOMSTD.101.2000014>
- [8] 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>
- [9] Intel Corporation, "Intel vRAN Accelerator ACC100 Adapter: Product Brief," Intel Corporation, Product Brief, Q4 2021. [Online]. Available: <https://cdrdv2-public.intel.com/642103/Intel%20vRAN%20Accelerator%20ACC100%20Adapter.pdf>
- [10] A. Ltd, "Xilinx T1 Telecoms Accelerator Card - What You Need to Know," *AccelerComm News*, Apr. 2025. [Online]. Available: <https://www.accelercomm.com/news/high-performance-accelerator-cards>
- [11] J. Boccuzzi, "Building Software-Defined, High-Performance, and Efficient vRAN Requires Programmable Inline Acceleration," *NVIDIA Developer Blog*, May 2023.
- [12] T. F. Staff, "The Impact of 5G and Cloud on Telco CapEx and OpEx," *TM Forum*, 2023. [Online]. Available: <https://inform.tmforum.org/features-and-opinion/the-impact-of-5g-and-cloud-on-telco-capex-and-opex>
- [13] IEEE ComSoc Tech Blog, "vRAN Market Disappoints – Just Like OpenRAN and Mobile 5G," January 2025. [Online]. Available: <https://techblog.comsoc.org/2025/01/07/vran-market-disappoints-just-like-openran-and-mobile-5g/>
- [14] K. Aarsal, "MWC 2025: Challenges of AI RAN as a Revenue Driver," *Omdia Analyst Opinion*, Mar. 2025. [Online]. Available: <https://omdia.tech.informa.com/om129572/mwc-2025-challenges-of-ai-ran-as-a-revenue-driver>
- [15] AI-RAN Alliance, "AI-RAN Alliance Vision and Mission White Paper," AI-RAN Alliance, White Paper, Dec. 2024. [Online]. Available: [https://ai-ran.org/wp-content/uploads/2024/12/AI-RAN\\_Alliance\\_Whitepaper.pdf](https://ai-ran.org/wp-content/uploads/2024/12/AI-RAN_Alliance_Whitepaper.pdf)
- [16] S. Kasi and K. Jamieson, "Towards Quantum Belief Propagation for LDPC Decoding in Wireless Networks," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '20. New

- York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: <https://doi.org/10.1145/3372224.3419207>
- [17] S. Kasi, P. Warburton, J. Kaewell, and K. Jamieson, "A Cost and Power Feasibility Analysis of Quantum Annealing for NextG Cellular Wireless Networks," *IEEE Transactions on Quantum Engineering*, vol. 4, pp. 1–17, 2023. [Online]. Available: <https://doi.org/10.1109/TQE.2023.3326469>
- [18] B. Villalonga, D. Lyakh, S. Boixo, H. Neven, T. S. Humble, R. Biswas, E. G. Rieffel, A. Ho, and S. Mandrà, "Establishing the Quantum Supremacy Frontier with a 281 Pflop/s Simulation," *Quantum Science and Technology*, vol. 5, no. 3, p. 034003, apr 2020. [Online]. Available: <https://dx.doi.org/10.1088/2058-9565/ab7eeb>
- [19] D-Wave Systems Inc., "Advantage Datasheet," 2022. [Online]. Available: [https://www.dwavequantum.com/media/htjclcey/advantage\\_datasheet\\_v10.pdf](https://www.dwavequantum.com/media/htjclcey/advantage_datasheet_v10.pdf)
- [20] S. Kim, S.-W. Ahn, I.-S. Suh, A. W. Dowling, E. Lee, and T. Luo, "Quantum Annealing for Combinatorial Optimization: A Benchmarking Study," *npj Quantum Information*, vol. 11, no. 1, p. 77, May 2025. [Online]. Available: <https://doi.org/10.1038/s41534-025-01020-1>
- [21] 3GPP, "3rd Generation Partnership Project; 5G; NR; Overall description; Stage-2," 3GPP TS 38.300 Release 18, May 2022.
- [22] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Radio Access Network; 5G; NR; Physical channels and modulation," 3GPP TS 38.211 Release 17, 2023.
- [23] H. Khedher, S. Hoteit, P. Brown, R. Krishnaswamy, W. Diego, and V. Veque, "Processing Time Evaluation and Prediction in Cloud-RAN," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6. [Online]. Available: <https://doi.org/10.1109/ICC.2019.8761870>
- [24] J. Ding, R. Doost-Mohammady, A. Kalia, and L. Zhong, "Agora: Real-Time Massive MIMO Baseband Processing in Software," in *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 232–244. [Online]. Available: <https://doi.org/10.1145/3386367.3431296>
- [25] 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>
- [26] S. Bhaumik, S. P. Chandrabose, M. K. Jataprolu, G. Kumar, A. Muralidhar, P. Polakos, V. Srinivasan, and T. Woo, “CloudIQ: A Framework for Processing Base Stations in a Data Center,” in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, ser. Mobicom ’12. New York, NY, USA: Association for Computing Machinery, 2012, p. 125–136. [Online]. Available: <https://doi.org/10.1145/2348543.2348561>
- [27] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Pérez, “Resource Sharing Efficiency in Network Slicing,” *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 909–923, 2019. [Online]. Available: <https://doi.org/10.1109/TNSM.2019.2923265>
- [28] A. Manousis, R. A. Sharma, V. Sekar, and J. Sherry, “Contention-Aware Performance Prediction For Virtualized Network Functions,” in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM ’20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 270–282. [Online]. Available: <https://doi.org/10.1145/3387514.3405868>
- [29] Q. Yang, X. Li, H. Yao, J. Fang, K. Tan, W. Hu, J. Zhang, and Y. Zhang, “BigStation: Enabling Scalable Real-Time Signal Processing in Large MU-MIMO Systems,” in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM ’13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 399–410. [Online]. Available: <https://doi.org/10.1145/2486001.2486016>
- [30] R. Gallager, “Low-density Parity-check Codes,” *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962. [Online]. Available: <https://doi.org/10.1109/TIT.1962.1057683>
- [31] R. Tanner, “A Recursive Approach to Low Complexity Codes,” *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, 1981. [Online]. Available: <https://doi.org/10.1109/TIT.1981.1056404>
- [32] J. Chen and P. Fossorier, “Density Evolution for BP-based Decoding Algorithms of LDPC Codes and Their Quantized Versions,” in *IEEE GLOBECOM ’02*, vol. 2, 2002, pp. 1378–1382 vol.2. [Online]. Available: <https://doi.org/10.1109/GLOCOM.2002.1188424>

- [33] J. Chen, R. Tanner, C. Jones, and Y. Li, “Improved Min-sum Decoding Algorithms for Irregular LDPC Codes,” in *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, 2005, pp. 449–453. [Online]. Available: <https://doi.org/10.1109/ISIT.2005.1523374>
- [34] D. Hocevar, “A Reduced Complexity Decoder Architecture via Layered Decoding of LDPC Codes,” in *IEEE Workshop on Signal Processing Systems, 2004. SIPS 2004.*, 2004, pp. 107–112. [Online]. Available: <https://doi.org/10.1109/SIPS.2004.1363033>
- [35] I. Gomez-Miguel, 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*, ser. WiNTECH '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 25–32. [Online]. Available: <https://doi.org/10.1145/2980159.2980163>
- [36] 3GPP, “3rd Generation Partnership Project; 5G; NR; Physical layer procedures for data,” 3GPP TS 38.214 Release 17, Jan. 2023.
- [37] Mihai *et al.*, “Digital Twins: A Survey on Enabling Technologies, Challenges, Trends and Future Prospects,” *IEEE Communications Surveys & Tutorials*, pp. 1–1, 9 2022. [Online]. Available: <https://doi.org/10.1109/comst.2022.3208773>
- [38] Fuller *et al.*, “Digital Twin: Enabling Technologies, Challenges and Open Research,” *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020. [Online]. Available: <https://doi.org/10.1109/ACCESS.2020.2998358>
- [39] ZSM, “Enablers for AI-based Network and Service Automation,” ZSM012 Group Specification, July 2022.
- [40] C. Fiandrino, G. Attanasio, M. Fiore, and J. Widmer, “Toward Native Explainable and Robust AI in 6G Networks: Current State, Challenges and Road Ahead,” *Computer Communications*, vol. 193, pp. 47–52, 2022. [Online]. Available: <https://doi.org/10.1016/j.comcom.2022.06.036>
- [41] W. Samek, T. Wiegand, and K. Müller, “Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models,” *CoRR*, vol. abs/1708.08296, 2017. [Online]. Available: <http://arxiv.org/abs/1708.08296>
- [42] K. Cyras, R. Badrinath, S. K. Mohalik, A. Mujumdar, A. Nikou, A. Previti, V. Sundararajan, and A. V. Feljan, “Machine Reasoning Explainability,” *CoRR*, vol. abs/2009.00418, 2020. [Online]. Available: <https://arxiv.org/abs/2009.00418>

- [43] W. Swartout, C. Paris, and J. Moore, “Explanations in Knowledge Systems: Design for Explainable Expert Systems,” *IEEE Expert*, vol. 6, no. 3, pp. 58–64, 1991. [Online]. Available: <https://doi.org/10.1109/64.87686>
- [44] S. Morita and H. Nishimori, “Mathematical Foundation of Quantum Annealing,” *Journal of Mathematical Physics*, vol. 49, no. 12, p. 125210, 12 2008. [Online]. Available: <https://doi.org/10.1063/1.2995837>
- [45] D. Bertsimas and J. Tsitsiklis, “Simulated Annealing,” *Statistical Science*, vol. 8, no. 1, pp. 10 – 15, 1993. [Online]. Available: <https://doi.org/10.1214/ss/1177011077>
- [46] A. P. Punnen, *The Quadratic Unconstrained Binary Optimization Problem: Theory, Algorithms, and Applications*. Springer International Publishing, 2022. [Online]. Available: <http://dx.doi.org/10.1007/978-3-031-04520-2>
- [47] K. Boothby, A. D. King, and J. Raymond, “Zephyr Topology of D-Wave Quantum Processors,” D-Wave Technical Report, Tech. Rep. 14-1656A-A, September 2021. [Online]. Available: [https://www.dwavequantum.com/media/2uznec4s/14-1056a-a\\_zephyr\\_topology\\_of\\_d-wave\\_quantum\\_processors.pdf](https://www.dwavequantum.com/media/2uznec4s/14-1056a-a_zephyr_topology_of_d-wave_quantum_processors.pdf)
- [48] E. Lobe and A. Lutz, “Minor Embedding in Broken Chimera and Derived Graphs is NP-complete,” *Theoretical Computer Science*, vol. 989, p. 114369, Mar. 2024. [Online]. Available: <http://dx.doi.org/10.1016/j.tcs.2023.114369>
- [49] J. Cai, W. G. Mcready, and A. Roy, “A Practical Heuristic for Finding Graph Minors,” 2014. [Online]. Available: <https://arxiv.org/abs/1406.2741>
- [50] V. Choi, “Minor-Embedding in Adiabatic Quantum Computation: I. The Parameter Setting Problem,” 2008. [Online]. Available: <https://arxiv.org/abs/0804.4884>
- [51] V. Choi, “Minor-embedding in Adiabatic Quantum Computation: II. Minor-universal Graph Design,” *Quantum Information Processing*, vol. 10, no. 3, pp. 343–353, jun 2011. [Online]. Available: <https://doi.org/10.1007/s11128-010-0200-3>
- [52] E. Parker and M. J. D. Vermeer, *Estimating the Energy Requirements to Operate a Cryptanalytically Relevant Quantum Computer*. Santa Monica, CA: RAND Corporation, 2023. [Online]. Available: <https://doi.org/10.7249/WRA2427-1>
- [53] B. Dzogovic, V. T. Do, B. Feng, and T. van Do, “Building Virtualized 5G networks Using Open Source Software,” in *2018 IEEE Symposium on Computer Applications Industrial Electronics (ISCAIE)*, 2018, pp. 360–366. [Online]. Available: <https://doi.org/10.1109/ISCAIE.2018.8405499>
- [54] O. Arouk and N. Nikaein, “5G Cloud-Native: Network Management & Automation,” in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and*

- Management Symposium*, 2020, pp. 1–2. [Online]. Available: <https://doi.org/10.1109/NOMS47738.2020.9110392>
- [55] N. M. Rekoputra, R. Harwahu, and R. F. Sari, “Performance Study of OpenAirInterface 5G System on the Cloud Platform Managed by Juju Orchestration,” in *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, 2019, pp. 211–216. [Online]. Available: <https://doi.org/10.1109/ISRITI48646.2019.9034647>
- [56] B. Chun, J. Ha, S. Oh, H. Cho, and M. Jeong, “Kubernetes Enhancement for 5G NFV Infrastructure,” in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, 2019, pp. 1327–1329. [Online]. Available: <https://doi.org/10.1109/ICTC46691.2019.8939817>
- [57] L. Bonati, P. Johari, M. Polese, S. D’Oro, S. Mohanti, M. Tehrani-Moayyed, D. Villa, S. Shrivastava, C. Tassie, K. Yoder, A. Bagga, P. Patel, V. Petkov, M. Seltser, F. Restuccia, A. Gosain, K. R. Chowdhury, S. Basagni, and T. Melodia, “Colosseum: Large-Scale Wireless Experimentation Through Hardware-in-the-Loop Network Emulation,” in *2021 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2021, pp. 105–113. [Online]. Available: <https://doi.org/10.1109/DySPAN53946.2021.9677430>
- [58] I.-S. Comsa, A. De-Domenico, and D. Ktenas, “QoS-Driven Scheduling in 5G Radio Access Networks - A Reinforcement Learning Approach,” in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–7. [Online]. Available: <https://doi.org/10.1109/GLOCOM.2017.8254926>
- [59] O. Naparstek and K. Cohen, “Deep Multi-User Reinforcement Learning for Distributed Dynamic Spectrum Access,” *IEEE Transactions on Wireless Communications*, vol. 18, no. 1, pp. 310–323, 2019. [Online]. Available: <https://doi.org/10.1109/TWC.2018.2879433>
- [60] J. Tan, L. Zhang, Y.-C. Liang, and D. Niyato, “Deep Reinforcement Learning for the Coexistence of LAA-LTE and WiFi Systems,” in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6. [Online]. Available: <https://doi.org/10.1109/ICC.2019.8761566>
- [61] P. Rost, S. Talarico, and M. C. Valenti, “The Complexity–Rate Tradeoff of Centralized Radio Access Networks,” *IEEE Transactions on Wireless Communications*, vol. 14, no. 11, pp. 6164–6176, 2015. [Online]. Available: <https://doi.org/10.1109/TWC.2015.2449321>

- [62] D. Bega, A. Banchs, M. Gramaglia, X. Costa-Pérez, and P. Rost, “CARES: Computation-Aware Scheduling in Virtualized Radio Access Networks,” *IEEE Transactions on Wireless Communications*, vol. 17, no. 12, pp. 7993–8006, 2018. [Online]. Available: <https://doi.org/10.1109/TWC.2018.2873324>
- [63] J. A. Ayala-Romero, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, A. Banchs, and J. J. Alcaraz, “VrAIIn: A Deep Learning Approach Tailoring Computing and Radio Resources in Virtualized RANs,” in *The 25th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom ’19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: <https://doi.org/10.1145/3300061.3345431>
- [64] Z. Qi, C.-H. Tung, A. Kalia, and T. Chen, “Savannah: A Real-time Programmable mmWave Baseband Processing Framework,” 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, p. 1736–1738. [Online]. Available: <https://doi.org/10.1145/3636534.3698843>
- [65] 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)*. Boston, MA: USENIX Association, Apr. 2023, pp. 405–417. [Online]. Available: <https://www.usenix.org/conference/nsdi23/presentation/gong>
- [66] srsRAN, “Open Source SDR 4G/5G Software Suite from Software Radio Systems (SRS),” GitHub, Aug. 1, 2022 [Online]. [Online]. Available: <https://github.com/srsran/srsRAN>
- [67] M. Kim, D. Venturelli, J. Kaewell, and K. Jamieson, “Warm-started Quantum Sphere Decoding via Reverse Annealing for Massive IoT Connectivity,” in *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, ser. MobiCom ’22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 1–14. [Online]. Available: <https://doi.org/10.1145/3495243.3560516>
- [68] M. Kim, A. K. Singh, D. Venturelli, J. Kaewell, and K. Jamieson, “X-ResQ: Reverse Annealing for Quantum MIMO Detection with Flexible Parallelism,” 2024. [Online]. Available: <https://arxiv.org/abs/2402.18778>
- [69] M. Kim, D. Venturelli, and K. Jamieson, “Leveraging Quantum Annealing for Large MIMO Processing in Centralized Radio Access Networks,” in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM ’19.

- New York, NY, USA: Association for Computing Machinery, 2019, pp. 241–255. [Online]. Available: <https://doi.org/10.1145/3341302.3342072>
- [70] S. Kasi, J. Kaewell, and K. Jamieson, “A Quantum Annealer-Enabled Decoder and Hardware Topology for NextG Wireless Polar Codes,” *IEEE Transactions on Wireless Communications*, vol. 23, no. 4, pp. 3780–3794, 2024. [Online]. Available: <https://doi.org/10.1109/TWC.2023.3311204>
- [71] A. Das Sarma, U. Majumder, V. Vaidya, M. G. Chandra, A. A. Kumar, and S. Pramanik, “On Quantum-Assisted LDPC Decoding Augmented with Classical Post-processing,” in *Parallel Processing and Applied Mathematics*, R. Wyrzykowski, J. Dongarra, E. Deelman, and K. Karczewski, Eds. Cham: Springer International Publishing, 2023, pp. 153–164. [Online]. Available: [https://doi.org/10.1007/978-3-031-30445-3\\_13](https://doi.org/10.1007/978-3-031-30445-3_13)
- [72] Y. Guo, H. Zeng, F. Xiong, T. Luan, Z. Zhang, and X. Wang, “Quantum Annealing with Post-processing of Maximum Likelihood for LDPC Decoding,” in *2024 5th Information Communication Technologies Conference (ICTC)*, 2024, pp. 168–172. [Online]. Available: <https://doi.org/10.1109/ICTC61510.2024.10601825>
- [73] U. Majumder, A. D. Sarma, V. Vaidya, and M. G. Chandra, “On Quantum-Enhanced LDPC Decoding for Rayleigh Fading Channels,” in *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*, 2022, pp. 462–467. [Online]. Available: <https://doi.org/10.1109/SEC54971.2022.00070>
- [74] M. Gramaglia, P. Serrano, A. Banchs, G. Garcia-Aviles, A. Garcia-Saavedra, and R. Perez, “The Case for Serverless Mobile Networking,” in *2020 IFIP Networking Conference (Networking)*, 2020, pp. 779–784. [Online]. Available: <https://ieeexplore.ieee.org/document/9142747>
- [75] “Cloud Native Computing Foundation landscape,” <https://landscape.cncf.io/>, [Accessed 05-May-2022].
- [76] 3GPP, “Architecture Enhancements for Control and User Plane Separation of EPC Nodes,” Technical Specification (TS) 23.214, 2016, v. 14.2.2.
- [77] 3GPP, “3rd Generation Partnership Project; LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures,” 3GPP TS 38.213 Release 17, Feb. 2024.
- [78] M. Uitto and A. Heikkinen, “Evaluating 5G Uplink Performance in Low Latency Video Streaming,” in *2022 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, 2022, pp. 393–398. [Online]. Available: <https://doi.org/10.1109/EuCNC/6GSummit54941.2022.9815703>

- [79] Y. Liu, T. Zhang, K. Jamieson, and Y. Xie, “Seeing Through the Fog: Empowering Mobile Devices to Expose and Mitigate RAN Buffer Effects on Delay-Sensitive Protocols,” 2025. [Online]. Available: <https://arxiv.org/abs/2507.00337>
- [80] F. Yi, O. Michel, H. Wan, and K. Jamieson, “Understanding the Impact of Cellular RAN-induced Delay on Video Conferencing,” 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, p. 1641–1643. [Online]. Available: <https://doi.org/10.1145/3636534.3697445>
- [81] M. Ghoshal, I. Khan, Z. J. Kong, P. Dinh, J. Meng, Y. C. Hu, and D. Koutsonikolas, “Performance of Cellular Networks on the Wheels,” in *Proceedings of the 2023 ACM on Internet Measurement Conference*, ser. IMC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 678–695. [Online]. Available: <https://doi.org/10.1145/3618257.3624814>
- [82] F. Yi, H. Wan, K. Jamieson, and O. Michel, “Automated, Cross-Layer Root Cause Analysis of 5G Video-Conferencing Quality Degradation,” 2025. [Online]. Available: <https://arxiv.org/abs/2505.14540>
- [83] K. Ashfaq, G. A. Safdar, and M. Ur-Rehman, “Comparative Analysis of Scheduling Algorithms for Radio Resource Allocation in Future Communication Networks,” *PeerJ Computer Science*, vol. 7, p. e546, 2021. [Online]. Available: <https://doi.org/10.7717/peerj-cs.546>
- [84] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.
- [85] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” 2019. [Online]. Available: <https://arxiv.org/abs/1509.02971>
- [86] G. Dulac-Arnold, R. Evans, P. Sunehag, and B. Coppin, “Reinforcement Learning in Large Discrete Action Spaces,” *CoRR*, vol. abs/1512.07679, 2015. [Online]. Available: <http://arxiv.org/abs/1512.07679>
- [87] M. Series, “Minimum requirements related to technical performance for IMT-2020 radio interface(s) Report ITU-R M.2410-0,” 2017.
- [88] S. D'Oro, M. Polese, L. Bonati, H. Cheng, and T. Melodia, “dApps: Distributed Applications for Real-Time Inference and Control in O-RAN,” *IEEE Communications Magazine*, vol. 60, no. 11, pp. 52–58, nov 2022. [Online]. Available: <https://doi.org/10.1109/2Fmcom.002.2200079>

- [89] Bonati *et al.*, “OpenRAN Gym: An Open Toolbox for Data Collection and Experimentation with AI in O-RAN,” in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE Press, 2022, pp. 518–523. [Online]. Available: <https://doi.org/10.1109/WCNC51071.2022.9771908>
- [90] 3GPP, “3rd Generation Partnership Project; Technical Specification Group Radio Access Network; 5G; NR; User Equipment (UE) radio transmission and reception; Part 1: Range 1 Standalone,” 3GPP TS 38.101 Release 16, 2020.
- [91] B. Rabenstein and J. Volz, “Prometheus: A Next-Generation Monitoring System (Talk).” Dublin: USENIX Association, May 2015.
- [92] M. T. Kawser, N. I. B. Hamid, M. S. Alam, and M. Rahman, “Downlink SNR to CQI Mapping for Different Multiple Antenna Techniques in LTE,” *International Journal of Information Engineering and Electronic Business*, 2012.
- [93] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [94] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.
- [95] M. T. Ribeiro, S. Singh, and C. Guestrin, ““Why Should I Trust You?”: Explaining the Predictions of Any Classifier,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 1135–1144. [Online]. Available: <https://doi.org/10.1145/2939672.2939778>
- [96] S. M. Lundberg and S. Lee, “A Unified Approach to Interpreting Model Predictions,” *CoRR*, vol. abs/1705.07874, 2017. [Online]. Available: <http://arxiv.org/abs/1705.07874>
- [97] J. Gou, B. Yu, S. J. Maybank, and D. Tao, “Knowledge Distillation: A Survey,” *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, mar 2021. [Online]. Available: <https://doi.org/10.1007%2Fs11263-021-01453-z>
- [98] S. Wachter, B. D. Mittelstadt, and C. Russell, “Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR,” *CoRR*, vol. abs/1711.00399, 2017. [Online]. Available: <http://arxiv.org/abs/1711.00399>
- [99] C. D. Wilen, S. Abdullah, N. A. Kurinsky, C. Stanford, L. Cardani, G. D’Imperio, C. Tomei, L. Faoro, L. B. Ioffe, C. H. Liu, A. Opremcak, B. G. Christensen, J. L. DuBois, and R. McDermott, “Correlated Charge Noise and Relaxation Errors in Superconducting Qubits,” *Nature*, vol. 594, no. 7863, pp. 369–373, 2021. [Online]. Available: <https://doi.org/10.1038/s41586-021-03557-5>

- [100] N. P. Breuckmann and J. N. Eberhardt, “Quantum Low-Density Parity-Check Codes,” *PRX Quantum*, vol. 2, no. 4, Oct. 2021. [Online]. Available: <http://dx.doi.org/10.1103/PRXQuantum.2.040101>
- [101] “IEEE Standard for Information technology-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” *IEEE Std 802.11ad-2012*, pp. 1–628, 2012. [Online]. Available: <https://doi.org/10.1109/IEEESTD.2012.6392842>
- [102] H. Zhang and J. Moura, “The Design of Structured Regular LDPC Codes with Large Girth,” in *IEEE GLOBECOM '03*, vol. 7, 2003, pp. 4022–4027 vol.7. [Online]. Available: <https://doi.org/10.1109/GLOCOM.2003.1258984>
- [103] J. Lu and J. Moura, “Structured LDPC codes for High-density Recording: Large Girth and Low Error Floor,” *IEEE Transactions on Magnetics*, vol. 42, no. 2, pp. 208–213, 2006. [Online]. Available: <https://doi.org/10.1109/TMAG.2005.861748>
- [104] X.-Y. Hu, E. Eleftheriou, and D. Arnold, “Regular and Irregular Progressive Edge-growth Tanner Graphs,” *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 386–398, 2005. [Online]. Available: <https://doi.org/10.1109/TIT.2004.839541>
- [105] Z. Albataineh, K. Hayajneh, H. Bany Salameh, C. Dang, and A. Dagmseh, “Robust Massive MIMO Channel Estimation for 5G Networks Using Compressive Sensing Technique,” *AEU - International Journal of Electronics and Communications*, vol. 120, p. 153197, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1434841119329358>
- [106] M. Tawada, S. Tanaka, and N. Togawa, “A New LDPC Code Decoding Method: Expanding the Scope of Ising Machines,” in *2020 IEEE International Conference on Consumer Electronics (ICCE)*, 2020, pp. 1–6. [Online]. Available: <https://doi.org/10.1109/ICCE46568.2020.9043057>
- [107] C. McGeoch and P. Farré, “Advantage Processor Overview,” D-Wave Systems, Tech. Rep., Jan 2022. [Online]. Available: [https://www.dwavesys.com/media/3xvdipcn/14-1058a-a\\_advantage\\_processor\\_overview.pdf](https://www.dwavesys.com/media/3xvdipcn/14-1058a-a_advantage_processor_overview.pdf)
- [108] T. V. Le, M. V. Nguyen, T. N. Nguyen, T. N. Dinh, I. Djordjevic, and Z.-L. Zhang, “Benchmarking Chain Strength: An Optimal Approach for Quantum Annealing,” in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 01, 2023, pp. 397–406. [Online]. Available: <https://doi.org/10.1109/QCE57702.2023.00052>

- [109] R. Ayanzadeh and M. Qureshi, “Skipper: Improving the Reach and Fidelity of Quantum Annealers by Skipping Long Chains,” 2023. [Online]. Available: <https://arxiv.org/abs/2312.00264>
- [110] D-Wave Systems, “Minorminer,” 2024. [Online]. Available: <https://github.com/dwavesystems/minorminer>
- [111] J. P. Pinilla and S. J. E. Wilton, “Layout-Aware Embedding for Quantum Annealing Processors,” in *High Performance Computing*, M. Weiland, G. Juckeland, C. Trinitis, and P. Sadayappan, Eds. Cham: Springer International Publishing, 2019, pp. 121–139.
- [112] S. Zbinden, A. Bärtschi, H. Djidjev, and S. Eidenbenz, “Embedding Algorithms for Quantum Annealers with Chimera and Pegasus Connection Topologies,” in *High Performance Computing*, P. Sadayappan, B. L. Chamberlain, G. Juckeland, and H. Ltaief, Eds. Cham: Springer International Publishing, 2020, pp. 187–206. [Online]. Available: [https://doi.org/10.1007/978-3-030-50743-5\\_10](https://doi.org/10.1007/978-3-030-50743-5_10)
- [113] 3GPP, “3rd Generation Partnership Project; Technical Specification Group Radio Access Network; 5G; NR; Multiplexing and Channel Coding,” 3GPP TS 38.211 Release 17, 2023.
- [114] T. Richardson and S. Kudekar, “Design of Low-Density Parity Check Codes for 5G New Radio,” *IEEE Communications Magazine*, vol. 56, no. 3, pp. 28–34, 2018. [Online]. Available: <https://doi.org/10.1109/MCOM.2018.1700839>
- [115] D. Divsalar, C. Jones, S. Dolinar, and J. Thorpe, “Protograph Based LDPC Codes with Minimum Distance Linearly Growing with Block Size,” in *GLOBECOM '05. IEEE Global Telecommunications Conference, 2005.*, vol. 3, 2005, pp. 5 pp.–. [Online]. Available: <https://doi.org/10.1109/GLOCOM.2005.1577834>

