ATHENA: Machine Learning and Reasoning for Radio Resources Scheduling in vRAN systems

Nikolaos Apostolakis, Marco Gramaglia, Livia Elena Chatzieleftheriou *(Member, IEEE)*, Tejas Subramanya, Albert Banchs *(Senior Member, IEEE)*, and Henning Sanneck

***Abstract*—Next-generation mobile networks will rely on their autonomous operation. Virtual Network Functions empowered by Artificial Intelligence (AI) and Machine Learning (ML) can adapt to varying environments that encompass both network conditions and the cloud platform executing them. In this view, it becomes paramount to *understand why* AI/ML algorithms made a decision, to be able to reason upon those decisions and, eventually, take further decisions related to *e.g.*, network orchestration. In this paper, we present ATHENA, an ML-based radio resource scheduler for virtualized Radio Access Network (RAN) system. Our real-software implementation shows that the proposed ML-based approach can outperform the baseline solution. We discuss how additional re-orchestration actions can be taken by analyzing our scheduling decisions and learning from the past.**

***Index Terms*—vRAN, Radio Resource Scheduling, Deep Reinforcement Learning, Machine Reasoning.**

I. Introduction

Several working groups in major Standard Development Organizations (SDO) are currently working on the autonomous operation of network components. Initiatives for including Artificial Intelligence (AI) and Machine Learning (ML) in the Radio Access Network (RAN) [1], Core [2], and Management [3] have been studied by the 3rd Generation Partnership Project (3GPP), while other initiatives led by the European Telecommunications Standards Institute (ETSI), such as the Experiential Networked Intelligence (ENI) [4] and Zero-touch Network and Service Management (ZSM) [5] groups, and industrial consortia such as O-RAN [6] are also promoting the introduction of data analytics into their architecture [7], [8], [9].

This integration will create control loops [10] among different network domains (*e.g.*, RAN / Core Management) to seamlessly operate mobile networks, harvesting data from the different available sources (*i.e.*, network functions, infrastructure), and enforcing back automated decisions into it. Integrating the application of data analytics tasks and intelligent algorithms into the network architecture could benefit the autonomous operation of the system, where components are self-organizing and self-orchestrating because it promotes the automation of decisions within the network. On the other hand, it also introduces other challenges that must be addressed.

N. Apostolakis, L. E. Chatzieleftheriou, and A. Banchs are with IMDEA Networks Institute and with the Department of Telematic Engineering, Universidad Carlos III de Madrid (UC3M), in Spain ({nikolaos.apostolakis, livia.chatzieleftheriou, albert.banchs}@imdea.org). M. Gramaglia is with the Department of Telematic Engineering, UC3M, Spain (mgramagl@it.uc3m.es). T. Subramanya and H. Sanneck are with Nokia Bell-Labs Germany ({tejas.subramanya, henning.sanneck}@nokia-bell-labs.com.)

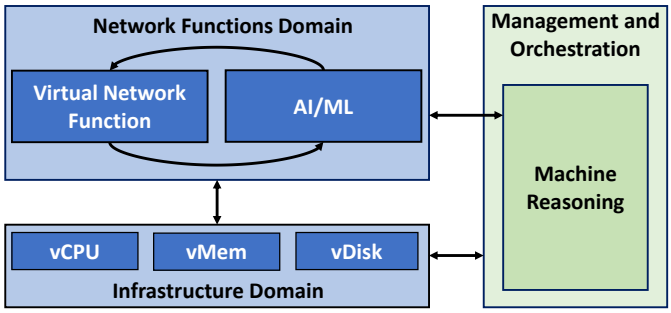

Fig. 1. Machine Reasoning in network management and orchestration.

Among them, we can list the scalability of the algorithms, especially for the radio components, and the *explainability* of the decisions taken by the intelligent algorithms.

In this paper, we provide an ML algorithm for the autonomous operation of the system which we use to discuss the explainability of its decisions, as it introduces a completely new paradigm in the operation of mobile networks. By introducing explainable intelligence [11], the usual black-box behavior of the AI/ML models is opened up, allowing operators to interpret the decisions of the deployed intelligent algorithms [12]. This concept has been recently extended into the *Machine Reasoning* (MR) [13], which tries to imitate human reasoning in an analytical way, further devising decisions based on the decisions taken by intelligent algorithms. While reasoning on top of an intelligent system has been proposed more than three decades ago [14], with the advent of modern computing systems (especially parallel ones, such as GPUs) these techniques can effectively be used to programmatically manage network intelligence.

We present an example of MR applied to AI / ML in the mobile network context in Fig. 1. By interpreting the decisions that an AI/ML module takes, an MR module located in the network orchestrator can *understand why* some decisions are taken, *e.g.*, due to lack of resources, and then *take further actions* by performing *e.g.*, a re-orchestration of the system. In this paper, we propose ATHENA (ArTificial intelligence-based scHEduler for radio resources with computiNg Awareness) as the fundamental building blocks of this view: an ML solution for the operation of a network function and the application of reasoning techniques stemming from the decisions of this algorithm. More in detail, our contributions are as follows:

- A novel ML-based radio controller for virtualized RAN (vRAN), that optimizes the network performance by

pairing with other components in the O-RAN ecosystem, such as the UE selection mechanism, while taking into account the computing resource impairments that are typical in cloud platforms [15].
- A reference implementation based on open source components such as `srsRAN` [16], extended with data shippers and new APIs for the integration of ML solutions into the baseline implementation, which we release as open source software.
- Reasoning on the decisions that are taken by the ML solution, and the subsequent analysis of these results towards the intelligent re-orchestration of the system.

This paper is structured as follows: we discuss the related work, especially from a network standardization perspective in Sec. II. We present the system model and the problem we study in Sec. III. Then we describe `ATHENA`, first by introducing the proposed ML-based radio scheduler in Sec. IV, and its implementation in Sec. V. Then we evaluate it in Sec. VI, and discuss the MR implications in Sec. VII, before concluding in Sec. VIII.

II. Context

In this section, we introduce the context of this work. We start with a discussion on automation and reasoning methods used in the standards (especially the ones related to O-RAN) and then discuss existing related works.

A. Automation and reasoning in the standards

1) 3GPP: There are various Working Groups (WG) within 3GPP that have currently active work items on the introduction of AI/ML within the network functionalities in the 3GPP framework. The RAN1 WG studies how to introduce AI/ML for the air interface, aiming to evaluate the performance of AI/ML-driven solutions compared to baseline solutions for channel state information feedback, beam management, and positioning [1]. The RAN3 WG studies AI/ML for NG-RAN, aiming to support AI/ML-related signaling within the existing NG-RAN architecture and for use cases such as network energy saving, load balancing, and mobility optimization. The SA2 WG studies how to incorporate AI/ML into the 5G core, aiming to further enhance the Network Data Analytics Function (NWDAF) for network automation [2] and a new 5G service to support application-level AI/ML operations [17]. SA5 WG studies the introduction of AI/ML management capabilities and services for 5G systems where AI/ML is being used, *e.g.*, Management Data Analytics Function and NWDAF [3].

2) O-RAN: Working Group 2 has released a technical report providing the terminology, workflow, and requirements related to AI/ML model training and deployment in the RAN, while also studying the enhancements needed in the architecture to support AI/ML-related requirements for several use cases such as QoE optimization and traffic steering [18].

3) ETSI: The ZSM ISG specifies new capabilities (*i.e.*, management services) and extends the existing ones in the ZSM architecture, to support automation of network management and orchestration operations based on AI/ML [5]. Finally, the ENI ISG is defining a cognitive network management architecture using AI/ML and context-aware policies to adjust offered services based on business goals by providing automated service provision, operation, and assurance [4].

In this work, we introduce as `ATHENA`: (i) an ML-based radio resource controller that takes into account the computing resource impairments to optimize radio resource utilization and the overall network performance, and (ii) an MR part, to extend the ML-based radio resource controller to also take further actions triggered by the explanation of the decisions taken by the ML model. This work is closely aligned with various SDOs because 3GPP and O-RAN are expected to study the ML-based radio resource scheduling use case in later releases while capabilities related to AI/ML explainability are already being studied and introduced in ETSI ZSM and ETSI ENI. We propose a reference deployment of `ATHENA` using the O-RAN architecture in Fig. 4, where the ML is part of `ATHENA` is an xApp in the Near-Real-Time Resource Intelligence Controller (Near-RT RIC), which operates at millisecond scale, while the MR module is part of the Non-Real-Time RIC (Non-RT RIC), which operates at minute granularity.

B. Related Work

1) Deep Reinforcement Learning-based radio schedulers: Deep Reinforcement Learning (DRL) is recently gaining attraction for solving problems in dynamic scenarios, as it does not require labeling new samples and training the model *ex novo*. In [19], the authors propose an actor-critic agent that chooses the most appropriate scheduling algorithm among the candidate scheduling algorithms to maximize the overall QoS. The state includes the number of active users, the arrival rate, the CQIs, and the performance measure in relation to the QoS requirements of users. The reward function measures the actual impact of choosing a rule on meeting the QoS objective for users. In [20], the authors propose a lightweight multi-user DRL algorithm to address the problem of spectrum access. For each time slot, a user can select only one channel to transmit, and an ACK signal (acts as an observation) is received by the user if the transmission is successful. The state includes the history of transmissions on the selected channel (*i.e.*, actions) and the observations made by the user. The reward function measures the achievable throughput. In [21], multiple access schemes are leveraged to divide the time frame amongst WiFi and LTE users. The authors propose a DRL algorithm to find the proper splitting point (*i.e.*, action) according to the received feedback on the channel status for earlier time frames. The state includes idle slots and successful transmissions, and the reward function measures the transmission time given to the LTE users without violating the throughput requirements of WiFi users. In [22], the authors specifically focus on the radio resource scheduling problem in the MAC layer with the objective of jointly optimizing the throughput and fairness. They proposed a DRL algorithm that works agnostically for various numerologies without any need for retraining when numerology changes. The objective of the agent is to select an active UE from a candidate set of UEs and allocate the available resource block group (RBG) to the chosen UE. The

state includes eligibility of UEs, data rate, and fairness while the reward function measures the throughput. In all these works, the state space includes the channel conditions and the reward considers the achieved UE throughput. Our work considers these metrics as well and explicitly designs ATHENA for targeting specifically vRAN systems, as we discuss in Sec. III. However, as we discuss next, ATHENA leverages ML to perform radio resource management in the context of fluctuating computing capacity in virtualized environments, such as the infrastructure that the O-RAN applications are running on.

2) CPU-aware radio schedulers: Due to the dynamic variations in the consumption of computational resources by the virtualized RAN in the O-RAN cloud infrastructure, there may be under-provisioning of computational resources, severely degrading the network performance, *e.g.*, due to radio frames not being decoded in time. There has been some research work to determine the usage of computational resources by virtual RAN functions [23]. Also, the authors in [24] analyzed the computational resource usage of virtual RAN functions using a real experimental testbed but without considering the impact of SNR on the computational load. The work [25] proposes a model to determine the computational load by considering the impact of SNR and the applied MCS on computational resource consumption. They illustrate that computational resource usage decreases when selecting more robust MCS schemes. This is leveraged in [26], where two main functions of the protocol stack are re-designed to avoid network performance degradation in virtualized RAN by avoiding computational outages disruption in the perceived performance, by optimizing per-frame the MCS and resource block assignment. However, the two latter works are leveraging models to provide their solutions. *In ATHENA, we use a model-free solution based on contextual bandit* and RL techniques.

Essentially, vrAIn project is the closest to the spirit of our work [27]: they proposed an architecture to control the CPU quota and maximum MCS in order to maximize the quality of service, in terms of users' buffer occupancy, jointly with low decoding error probability and minimum computational resources. Similar to ATHENA, they solve a contextual bandit problem with RL. It is leveraging the channel statistical characteristics, while, as we explain later, ATHENA takes as input the instantaneous channel condition and outputs the exact MCS and PRB allocation to increase the user's throughput, receiving direct feedback from an internal module of the vRAN (LDPC decoder). While vrAIn operates at coarse time scales (in the order of seconds) and fits in the Non-RT RIC of the O-RAN architecture, ATHENA is forming a millisecond-order loop at the MAC layer and, thus, is placed in the O-DU and/or Near-RT RIC.

3) Reasoning and explainability in ML and AI: AI and ML interpretability is a new topic that emerged recently and is currently very active [28], [11]. The interest in ML and AI explainability is to evaluate a learned model and to also help its users to trust its decisions. However, although AI and ML methods are usually evaluated over specific metrics, there is not yet a consensus on the definition of their *interpretability* and *explainability*, although the problem is being studied by different disciplines [29]. In general, it is still unclear what the characteristics that the interpretation of a black-box model should have in order to produce outputs that are relevant to a specific audience or problem [30].

In this work we propose the idea of *actionable* MR [13], [31], that is, taking additional decisions based on the ones already taken by the AI/ML. This approach suits very well the management and orchestration of a 5G network, as this task is usually happening at a different time-scale [32] from other AI/ML-based operations. To the best of our knowledge, this is the first work that introduces the concept of actionable MR in the O-RAN ecosystem. We leverage these techniques to analyze the behavior of the underlying AI/ML rApps/xApps/dApps [33] and to re-orchestrate resources to meet operator-defined requirements.

TABLE I
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	
TTI	Transmission Time Interval	t		
		β	Congestion factor	$\mathcal{B}$

III. SYSTEM MODEL AND PROBLEM

In this section, we present our model for radio scheduling, then the resource scheduling at virtualized RAN, and finally, we discuss how decoding is performed and its complexity.

A. System model

We focus on the physical (PHY) and the MAC layers of the 3GPP 5G-NR stack [34]. These layers take care of several functionalities, ranging from very low PHY level functionality (*e.g.*, demodulation and decoding) to radio resource scheduling to UEs. We depict our model in Fig. 2 and provide notation in Table I.

1) Model components: Our system consists of a set $\mathcal{U}$ of user equipment (UEs) u and a base station, which we will also call gNB, that provides communication capabilities to users. The UEs communicate with the gNB in order to transmit their information, according to the radio resource scheduling policies defined by the gNB. These policies are effectively translated into a specific amount of Physical Resource Blocks (PRBs) and a specific Modulation and Coding Scheme (MCS), that we will next present in detail.

At each Transmission Time Interval (TTI) t, the gNB allocates to each UE u a number of PRBs and a specific MCS for its data transmission, both in Uplink (UL) and Downlink (DL). We focus on the UL pipeline, because this creates a major computing load for the gNB [24] and, as we discuss in Sec. III-C, it is one of the most important factors to consider for vRAN systems.

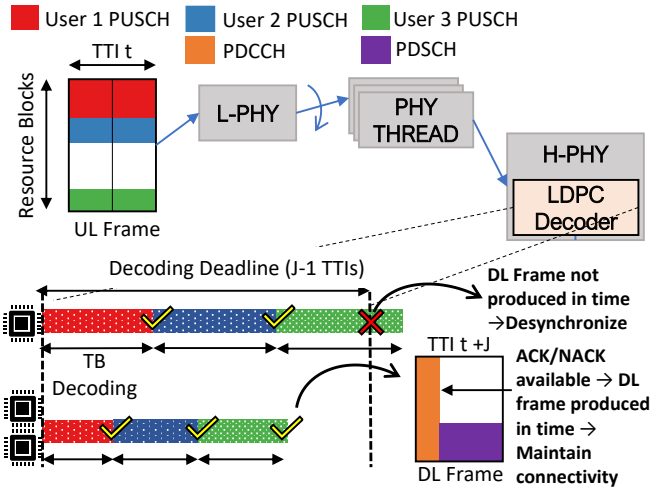

Fig. 2. Decoding in 5G-NR. The received UL frame is directed to the LDPC decoder, which decodes the users' TBs. i) Low capacity/highly congested CPUs: Total decoding time exceeds the deadline and the ACK/NACK is not integrated into the DL frame in time, incurring users' desynchronization. ii) High capacity/low congested CPUs: Decoding tasks end before the deadline and the information regarding ACK/NACK is packed in the PDCCH of the DL Frame, maintaining connectivity.

2) Physical Resource Blocks (PRBs): The total available bandwidth managed by the gNB is partitioned into N blocks, called Physical Resource Blocks (PRBs). Fifth-Generation New Radio (5G-NR) and Long Term Evolution (LTE) systems use an Orthogonal Frequency-Division Multiple Access (OFDMA) scheme for data transmission, encoding data allocated to a UE into a portion of such N blocks [34].

In 5G-NR systems, the number N of available PRBs is variable and depends on the total available bandwidth and the width of each PRB. For example, in a scenario with 10 MHz bandwidth and 180 KHz-wide PRBs, there are $N_{RB}^{UL} = 50$ available PRBs during every TTI. However, since 5 PRBs must be reserved for the Physical Uplink Control Channel (PUCCH) [35], the maximum number of PRBs that is actually available for the PUSCH is $N = 45$. Also, the assigned number n_u of PRBs for the PUSCH of user u must be factorizable into factors of 2,3 and 5 [35], *i.e.*, $n_u = 2^{\alpha_0} \cdot 3^{\alpha_1} \cdot 3^{\alpha_2} \leq N$, where $\alpha_0, \alpha_1, \alpha_2$ non-negative integers. Thus, at each TTI, the gNB should decide how to assign n_u PRBs to UE u. The user u is selected at each TTI by the MAC scheduler.

3) Modulation Coding Scheme (MCS): Each user's u Modulation Coding Scheme (MCS) m_u belongs to a set $\mathcal{M} = \{1, ..., M\}$ of M elements, where each element captures a number of bits that will be encoded in each symbol transmitted by user u. For 5G-NR, 3GPP defines 29 MCS indexes, *i.e.*, $M = 29$, 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. Thus, in practice, not all MCS indices are always available.

4) Radio Resource Scheduling Procedure: With the term *radio resource scheduler*, we define the component that resides on the MAC layer of the O-DU and performs the following three tasks; i) user selection, ii) PRB allocation, and iii) MCS selection. The UL radio resource scheduling is performed by a centralized agent in the gNB, *e.g.*, the network operator, and consists in deciding the user u granted data for the UL channel at every TTI t. Periodically, each user u 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 UE's u stack. Then, the gNB schedules the user u transmission and answers with a *transmission grant* for user u. The grant consists of two decisions: (i) the number n_u of PRBs that are assigned to user u, and (ii) the MCS m_u that u must use. The choice of the granted user $u \in \mathcal{U}$, and the selection of n_u and m_u 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.

5) Transport Block Size (TBS): The selection of MCS and PRBs eventually leads to a given amount of data that can be transmitted by a UE, which is called Transport Block (TB) and has a specific size (TBS) l_u that accounts for the user data, the size of the extra information such as Cyclic Redundancy Checks (CRCs), 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].

B. Problem: Resource scheduling in vRAN

Virtualized RANs (vRANs) are a successful network paradigm, as they allow to decouple the underlying network infrastructure from the software-based implementation of a 3GPP gNB. Hence, vRANs have become a very important technology for next-generation mobile networks, also driven by associations such as O-RAN [6]. However, the operation of vRAN systems still presents some fundamental problems, such as the integration of hardware accelerators or the optimal interplay between the cloud infrastructure and the vRAN software. In this paper, we aim at maximizing the system throughput in the communication between the end users and the gNB, considering the possible impairments due to the cloud infrastructure. We take a step further from the direction identified in [37] and consider a deep learning approach.

According to the 3GPP 5G-NR specification [35], each UL/DL frame transmission is associated to 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 to manage the whole data transmission's lifecycle. After the UL frame receival by the Low-PHY (L-PHY), a PHY-thread starts processing and directs it to the High-PHY (H-PHY) layer containing the decoder module. We describe in Sec. III-C the role of PHY-thread. The decoder will start sequentially decoding the TB of each user and will inform the corresponding HARQ process about the ACK/NACK of the UL-transmitted 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 in order to increase the probability of successful decoding. However, all the tasks that are related to uplink (UL) frame decoding (at the PHY layer) have to be completed before the downlink (DL) frame, containing the ACK/NACK, is scheduled to be sent. The ACK/NACK is contained in the Physical Downlink Control Channel (PDCCH), located in the first symbols of the DL frame, while the rest symbols transmit the downlink data in the Physical Downlink Shared Channel (PDSCH). This introduces a hard deadline for the completion of the decoding tasks (that are by far the most computing expensive of the full protocol stack [38], [39], [15], [24]). In 5G-NR this deadline is configurable and let it be equal to $J-1$ Transmission Time Intervals (TTIs). That is, the frame decoding result has to be ready before the ACK/NACK transmission that will take place in the J-th 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) impose a deadline of 3 ms to complete de-modulation, rate-matching, and decoding operations [38]. In Fig. 2, we depict the UL frame decoding procedure under two scenarios; i) CPUs that are congested with third-party external tasks (also referred to as low capacity CPUs), for which decoding deadline violations, and thus user desynchronization, occur and ii) uncongested CPUs (*i.e.*, with high available capacity), where the frame can be decoded 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 [15], 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 [40], 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 [41]: 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 some UL frames, can cause the UE to completely lose the synchronization, and hence the associated data flow [15]. This would cause additional congestion, as the same packets will need to be sent again.

In Sec. IV we propose an ML solution that jointly optimizes the UL decisions and the computing effort generated by them. This avoids disruptions that may be caused when the generated computing load is too high, *e.g.*, due to decisions that impose computing-hungry operations at the same time. We next 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.

C. Decoding in vRAN systems

1) Decoding algorithm: In a 3GPP PHY pipeline, a new decoding task takes place at each TTI. Decoding tasks translate the modulated and encoded digital symbols that are received in the Physical Uplink Shared CHannel (PUSCH), which carries UL data, into bytes associated with each UE that are delivered to the higher layer of the gNB stack.

After de-modulating the symbols and performing other channel optimization procedures, each bit belonging to the original transport block is decoded using an iterative procedure. During this process, each bit i is represented as a value LLR_i in the range $[-1 \dots 1]$, carrying the log-likelihood ratio of the i-th symbol being either a 0 or a 1. That is, the decoder processes a vector LLR of size TBS, containing the log-likelihood ratios for all the bits in the TB. When $LLR_i \to 1$, then likely bit $i = 1$, while $LLR_i \to -1$ means that likely bit $i = 0$. Finally, when $LLR_i = 0$ there is high uncertainty on the final value of the decoded bit i.

The decoder algorithm loops across all the LLR vector items LLR_i to maximize the likelihood of the received bits in the TB, and repeats the loop until a threshold likelihood ratio is achieved. After that, the TB is considered decoded and the CRC is checked, to discover possible channel errors. Standards define specific instances of this decoding process: the Low-Density Parity Check (LDPC) decoder is used in 5G-NR and the Turbo decoder in LTE [42]. Both the LDPC and the Turbo decoder share the generic iterative procedure that we described in this section.

2) Software implementations and complexity: The implementation of the procedure discussed above can be represented by two nested loops: an inner one that iterates over the LLR_i 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 time complexity 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 (*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 we 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 with one iteration 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 re-transmission. This value is implementation-dependent. In the software implementation used in this paper $I_{MAX} = 8$.

An LDPC/Turbo decoder instance is present in every PHY layer thread (namely DSP worker in Fig. 2) that handles UL frame processing in each TTI. Since the entire UL processing pipeline can typically take longer than a single TTI, 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` [43] deploys 3 PHY-layer threads by default. If an UL frame is received and all threads are occupied, the UL pipeline pauses

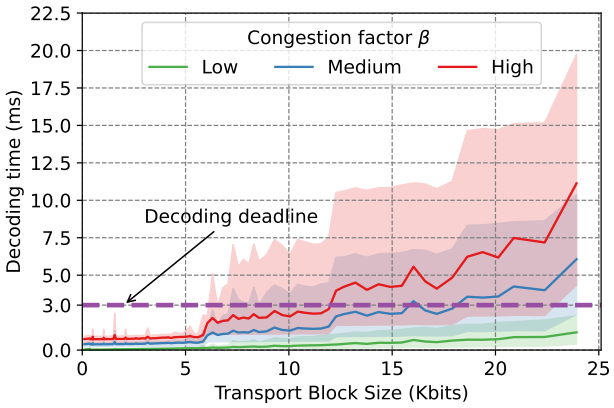

Fig. 3. Decoding time T_{dec} for MCS $\in [0, 24]$, various eligible numbers of UL PRBs, under different SNR conditions, and three congestion factor β. Solid lines denote the averages, while shaded areas depict the 5th to 95th percentile.

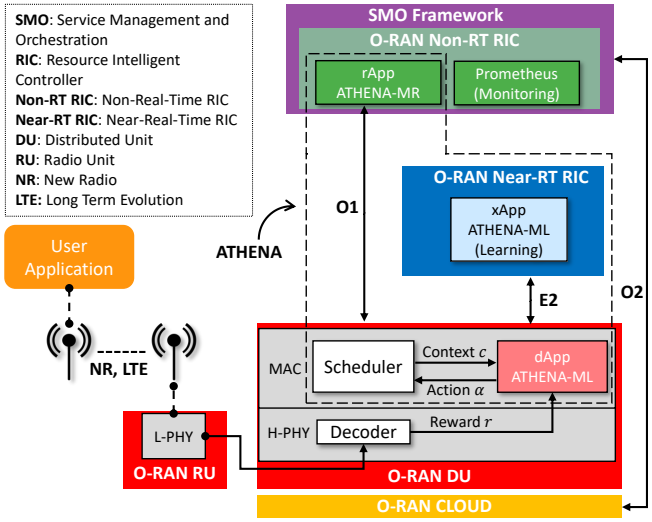

Fig. 4. ATHENA and its integration with the most important modules of the 3GPP and O-RAN architectures. ATHENA framework comprises i) ATHENA−ML resource controller that integrates with a user selection component to provide a fully-fledged MAC scheduler, and ii) ATHENA−MR engine that (re-)orchestrates O-Cloud resources to optimize the O-DU behavior.

until a thread becomes available. Furthermore, if multiple users are scheduled in a TTI, the decoder of the respective PHY thread processes their transport block sequentially, after which the UL pipeline resumes. As a result, if the total decoding time exceeds the deadline, all connected users will become desynchronized from the network and experience a degradation in the quality of service.

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. 3 shows T_{dec} *vs.* TBS (obtained with the platform discussed in Sec. VI).

It illustrates the elapsed time for full decoding, under varying channel quality conditions, and different competing loads (modeled by the variable β, which we will later introduce).

When no competing load is present (green line in Fig. 3), 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 $J - 1$ TTIs, which in 5G systems is 3 ms for mobile broadband traffic.

3) 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 quantum 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, we introduce a *congestion factor* $\beta \in [0, 1]$ that mimics the overhead introduced 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.

To indicate that there are no competing processes and that we can operate at a full CPU speed we assume $\beta = 0$, while $\beta = 1$ indicates a very slow and crowded computing platform. More details about the implementation of β are provided in Sec. V. Congestion factor β takes into account the dynamic nature of the data center the O-RAN cloud is running on. That is, this is not apriori fixed and can be by preference configured by an orchestration algorithm, as we show in Sec. VII-C, or be tracked and measured, as we explain in Sec. V-B7.

IV. ATHENA

We now discuss ATHENA framework, which as depicted in Fig. 4, 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's 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 millisecond timescale [34] providing uplink scheduling grants to UEs. For the rest of the paper, we consider that any user selection component could be integrated in ATHENA (*e.g.*, round-robin, proportional fair) and we study the radio resource controller component. In this section, we focus on ATHENA−ML, while ATHENA−MR is studied in Sec. VII.

ATHENA−ML, a contextual bandit 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 β. ATHENA−ML

then transforms this information into policies, optimizing RAN performance to the hard time deadlines imposed by the computing platform.

We formulate our problem as a contextual bandit (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 our 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 γ, the reward in CB equals the instantaneous collected reward. Therefore, any CB problem can be solved using RL algorithms considering 1-step episodes or alternatively $\gamma = 0$.

ATHENA-ML uses an RL algorithm to solve the CB problem by adjusting it to 1-step episodes. The resulting policy function $\pi(c) : C \rightarrow \mathcal{A}$ 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. ATHENA-ML is also 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.

A. ATHENA-ML components

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 as we discuss in detail in Sec. V-B6. 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. We define $c^t := (\beta^t, \bar{c}_u^t)$ to be the context vector representing the system's context at stage t. We define the context space $\mathcal{C} := \mathcal{B} \times \mathcal{R}$.

2) The action space $\mathcal{A}$: As existing radio resource management works [44], 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 $m_u \in \mathcal{M} := \{1, \ldots, M\}$ be the MCS decision for user u, where $\mathcal{M}$ is a set of possible MCS that is available by the 5G radio technology.

Thus, we define the action space $\mathcal{A} := \mathcal{N} \times \mathcal{M}$, essentially capturing all possible pairs of decisions m_u^t and n_u^t allocated to user $u \in \mathcal{U}$ in decision episode t.

The decision/action is transferred to the user and modulates the UL frame accordingly. The environment is the computational platform and more specifically the LDPC decoder, which decodes the UL frame and gives back the reward signal.

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, we move from the discrete nature of the decision variables $\alpha_u^t = (n_u^t, m_u^t)$ to the continuous one $\hat{\alpha}_u^t = (\hat{n}_u^t, \hat{m}_u^t)$. By doing so, we simplify the ML task 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 we discretize to space $\mathcal{N} \times \mathcal{M}$ according to the procedure we will describe later. By moving to the continuous space, we manage to associate adjacent actions which can not happen in the discrete space where every action is distinct.

4) Reward function: Given the $\bar{c}_u$ condition for each user u and the congestion factor β, we encourage the actions α that will probably lead to successful decoding, ultimately pushing the system to learn allocations that result in high system throughput[1].

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.

We design ATHENA-ML's reward function to account for this observation. Our reward captures the contribution of (i) the data bits d_u^t that user u is granted by the gNB to transmit under action α_u^t; (ii) a binary variable r_u^t denoting the result of the Cyclic Redundancy Check (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. As we will discuss in Sec. VI, ATHENA-ML learns quickly, because it takes advantage of both CRCs' values *and* decoding time.

B. ATHENA-ML internal design

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 [45], which has shown superior scalability properties [45, Chapter 13.1]. AC belongs to the policy gradient family of algorithms, can operate on

[1]This assumption makes ATHENA-ML better suited for enhanced Mobile BroadBand (eMBB) scenarios, as we do not take into account other metrics that could be useful for e.g. low latency scenarios such as the guarantees on the maximum length of UE transmission queues.

continuous-valued control spaces, and approximate directly the best policy function π 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, etc.), operate on distinct control spaces, approximate the state-value or the action-value function, and then apply an ϵ-greedy selection policy on the control space. This becomes intractable in big action spaces, such as in large bandwidth 5G deployments.

We opted for the Deep Deterministic Policy Gradient (DDPG) [46] algorithm due to its benefit to resolve 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 α in context c. The critic is represented by a neural network $Q_\phi(c, \alpha)$, parameterized by weights ϕ. Given a set of interactions $\mathcal{D}$ consisting of samples (c, α, r), it minimizes the residual Mean Squared Error (MSE) between the predicted $Q_\phi(c, \alpha)$ and the received reward r:

$$\mathop{\mathbb{E}}_{(c,\alpha,r)\sim\mathcal{D}}[(Q_\phi(c,\alpha) - r)^2] \tag{1}$$

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

$$\mathop{\mathbb{E}}_{c\sim\mathcal{D}}[Q_\phi(c, \mu_\theta(c))] \tag{2}$$

2) Actor inference: As we mentioned in Sec. IV-A3, for scalability reasons, we rescaled the output action 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 $\hat{\alpha}$, we implemented the following hierarchical blocks [47]:

1) **Action Generation**: We applied the K-Nearest Neighbors (k-NN) function $g_k(\hat{\alpha}) = \underset{\alpha\in\mathcal{A}}{\overset{k}{\arg\min}}|\alpha - \hat{\alpha}|_2$, 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, we evaluate each of the k actions and select the highest rewarding one according to the prediction of the critic.

$$\pi_{\theta,\phi}(c) = \underset{\alpha\in g_k\circ\mu_\theta(c)}{argmax}\ Q_\phi(c, \alpha) \tag{3}$$

3) Application to O-RAN and 5G-NR standards: As depicted in Fig. 4, 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 TTI. We require this location 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 [48]. Frequent user scheduling, at every TTI to provide high throughput and low latency, and high 5G high bands incur high SNR variations within very few TTIs, as discussed in [49]. This requires rapid reactions from the gNB side to maintain high throughput, by achieving decodability and staying within the deadline constraints. We place ATHENA-ML 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-RT RIC, decoupled from the user plane.

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

We leverage the independent lifecycle of each HARQ's data transmission and demultiplex the DDPG agent into $|H|$ agents, as many as the HARQ processes. We name them *HARQ agents* and they share the same parameters θ, ϕ. Each agent i interacts with the HARQ process i. At each TTI t, we assign the agent indexed by $t \mod |H|$ 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, we fit them in the Distribution Unit (DU) of the O-RAN architecture.

We also offload the learning functions of the HARQ agents to a *main* agent. The main agent shares the same model parameters with the HARQ agents. At every scheduling opportunity, the HARQ agents store the samples, consisting of context, action, and reward (c, α, 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 or Non-RT RIC. In Fig. 4, we depict how ATHENA-ML integrates with the O-RAN architecture. We have placed the main agent running as a xApp in the Near-RT RIC and the HARQ agents running in O-DU as *dApp*, adopting the concept from [33].

4) Offline Training: O-RAN principles require *pretraining* of the agent models before they are deployed on a live production environment [50]. To comply with this requirement, we collected a dataset $\mathcal{D}$ from an isolated sandboxed environment multiple samples for randomly taken decisions $\alpha_t = (n_t, m_t)$ and contexts $c_t = (\beta_t, \bar{c}_t)$ where we recorded the decoding

time T_{dec} and the CRC result r as returned by the LDPC decoder. In order to accelerate training, reduce the sample complexity, and remove unavoidable outliers produced by a real system, we grouped per $(\beta, \bar{c}, n, m)$ and modified the returned reward for each group as follows:

$$R^{group} = \begin{cases} d_u, & P[r=1] \geq r_{thres} \text{ and } T^{\gamma}_{dec} \leq J-1 \\ -K, & \text{otherwise} \end{cases} \tag{4}$$

where T^{γ}_{dec} is the γ-percentile $\in [0,1]$ of T_{dec} within the group and we consider it 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 γ would lead to learning more conservative policies, because of the negative reward, and vice-versa. With r_{thres}, we set a target threshold value above which we can consider satisfactory data transmission, in terms of successful error correction.

V. Implementation

To prove the feasibility of ATHENA, we implemented and integrated it on srsRAN [43], an open-source software framework that implements the functionality from the PHY up to the higher layers for eNB/gNBs that is compliant with the major SDO architectures. We used the 22.04 version of the software [16], which provisions the Rel. 15 of the 3GPP standard, and implemented ATHENA in a compliant way with the O-RAN reference architecture (see also Sec. II). The software is written in C/C++, and it is one of the most important tools for open experimentation with LTE and 5G prototypes. While srsRAN provides the software solution only, it relies on Software Defined Radio (SDR) cards for the RF frontend. For the implementation of ATHENA-ML[2]., however, we could not directly rely on this default implementation of srsRAN, for the reasons that we will next discuss.

A. ATHENA-ML integration into vRAN software

From Sec. III, 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, we need to span over very different channel conditions. While this scenario can be obtained with antennas and over-the-air transmission, we decided to not use this way for two reasons: first, we could not use licensed bands for our experiments, and, most importantly, using real mobile phones as UE would have limited the repeatability of the results as it required a non-negligible human intervention during the experiment (*i.e.*, to move the terminals creating different channel conditions).

Instead, we relied on a feature of srsRAN, namely the transmission of the modulated samples that would have been transmitted using hardware RF-frontend, using an alternative software RF-frontend based on ZeroMQ, an open source message queueing library written in C. When using this driver, the transmitted I/Q baseband symbols between UE and base station are transferred over various transport methods, like Inter-Process Communication or TCP sockets.

The use of ZeroMQ enables running the network in a fully softwarized way, and it also opens the opportunity of emulating complex network 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 [51]). Hence, to allow the programmability of our experiments we utilized GNU Radio Companion (GRC), which is another open-source project mainly used for SDR, and, among others, contains modules for ZeroMQ library. We coded in Python a GRC Broker, located between the UE and the cells of the gNB implementing the RF interface. This module intercepts the transmitted I/Q symbols and performs operations on them to emulate the channel conditions: to adjust the perception of the signal strength, we add multiplier boxes to each UE that time-domain cells' transmitted samples with 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 our experiments, we dynamically control G between .05 and 1 to emulate different channel conditions. Finally, to simulate an Additive White Gaussian Noise (AWGN) channel, we add an additive noise process to the intercepted symbols, which samples from a normal distribution with zero mean. More details about the implementation setup can be found in [52].

Then, we implemented ATHENA-ML using Python and Tensorflow. We integrated the implementation of ATHENA-ML directly in srsRAN scheduler, 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 our implementation either directly, as in the case of the SNR, or indirectly, as the β factor is sent to the controller by the monitoring application, as discussed in Sec. V-B7. ATHENA-ML has one *coordinator* process, responsible for directing requests and the decoding results to the corresponding worker agent based on the TTI, 8 HARQ agents' processes, as many as the HARQ processes. Since no online training is taking place on the experiments, we do not deploy the main agent process 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*.

As discussed in Sec. III, we model with β all the possible interfering factors for the UL frame decoding times. Since this implies an additional delay, we emulate β by introducing an additional workload at the end of each TB decoding iteration: We introduce $\beta_n = 1000 \cdot \beta$ square root computations that

[2]All the software components related to ATHENA-ML are available at https://github.com/kaposnick/athena_agent

will force the reduction of the effective capacity offered by the cloud platform.

B. 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, we detail them next.

1) Dataset Collection: To collect the training dataset $\mathcal{D}$, we 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 β and $\bar{c}$, we developed an automated process that sets the gain G of the channel and the congestion conditions on which the decoder threads are running.

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. 3. The critic's neural network comprises also fully connected layers with ReLU activation and one single output neuron, outputting the reward prediction, with linear activation.

3) Pretraining: Given the dataset $\mathcal{D}$, we grouped according to the procedure in Sec. IV-B4. We firstly pretrained the critic's neural network Q_ϕ as a normal regressor that minimizes the critic's objective, using Eq. 1. Successively, we froze the critic's weights and pretrained the actor's neural network μ_θ so that it maximizes its objective, *i.e.*, the critic's reward prediction, using Eq. 2. We pretrained both actor and critic networks using the Adam optimizer, with a learning rate $1\epsilon^{-4}$ for a period of 200 epochs. In turn, we deploy the agent's weights θ, ϕ on ATHENA-ML on the described testbed, which we infer using Eq. 3 using $k = 5$.

4) Scalability enhancements: While the srsRAN's ZeroMQ RF-frontend driver allows us to test ATHENA in a real system, its capabilities are mostly targeting the debug of the higher layers of the RAN stack. Hence, one of these limitations is the native support to 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, in order to overcome this and simulate a multi-user cell scenario, we designed a *digital twin* (DT). 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 our 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 DT, eliminating the restrictions of srsRAN's front-end. We define $\tau : (c, \alpha) \rightarrow (\hat{t}_{dec}, \hat{r})$ as the digital twin that maps the context c and action α to the expected decoding time $\hat{t}_{dec}$ and the decoding success probability $\hat{r}$. We designed the DT as a deep neural network that performs the following two prediction tasks; i) the decoding time prediction task, which yields the expected decoding time, ii) the decoding success probability prediction task, which computes the probability $P[r = 1]$. The DT has two output neurons, one for each prediction task; the decoding time prediction neuron, activated by the *linear* function, and the decoding probability prediction neuron, activated by the *sigmoid* function, which is typically used for classification tasks.

We trained the DT using collected data from the physical system in a supervised way using the Mean Squared Error (MSE) loss function for the decoding time prediction task and Binary Cross Entropy (BCE) loss for the decoding success probability task, which is negative of the log of corrected predicted probabilities. Additional information about the implemented DT can be found in [53].

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, we adjust ATHENA-ML's decision so that the equivalent TBS fills the requested demand. We also keep the new MCS less than ATHENA-ML's decided MCS in order to maintain the decodability of the frame.

6) SNR estimation: We adopt 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 wide band 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]), we act conservatively and take 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 [54, 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.

As we consider the SNR a proxy for the channel conditions,

we estimate the channel condition as:

$$\bar{c} = SNR + 10 \cdot log(PRB) \quad (5)$$

where $\bar{c}, SNR$ are expressed in dB. The variable $\bar{c}$ effectively measures the total SNR and Eq. 5 yields the same $\bar{c}$ for the same channel response G irrespective of the number of PRBs.

7) CPU Congestion Factor β: O-RAN Clouds (O-Cloud) allow 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, etc.), but, in principle, tenants are agnostic of the actual silicon their application is running on. The CPU congestion factor β, which effectively expresses the number of available CPU cycles that are allocated on the applications, can be directly configured by popular cloud orchestrators (such as `Kubernetes`) or specific orchestration algorithms, such as `ATHENA-MR` which we introduce in Sec. VII-C. Additionally, O-Cloud is enhanced with monitoring applications (*e.g.*, `Prometheus` [55]), 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 run on the Service Management and Orchestration Framework (SMO) of the O-RAN architecture in the Non-RT RIC. Independent of whether β 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 with respect to the infrastructure contextual fluctuations. In Fig. 4, we show how SMO monitors the O-Cloud infrastructure via the O2 interface and informs `ATHENA-ML` via the O1 interface.

C. Multi-user scenario

`ATHENA-ML` radio resource controller works on a single-user basis, allowing to select the optimal MCS and PRB combination at every point in time. As already discussed, `ATHENA` can integrate with different UE selection procedures. We picked `srsRAN`'s round-robin and matched it to `ATHENA-ML` that selects the best PHY layer parameter. Hence, for each TTI, 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 discussed in Sec. V-B4. We studied the integration of `ATHENA-ML` with the round-robin user selection procedure that imposes a maximum boundary on the number of selected users at each TTI 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 we discussed in this paper.

VI. Evaluation Results

In this section, we evaluate `ATHENA-ML` against the vanilla radio resource controller available in `srsRAN`, namely `Baseline`, which we consider as a general benchmark for commercial state-of-the-art radio resource managers. In order to demonstrate the need for having a joint PRB-MCS selection, we 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.

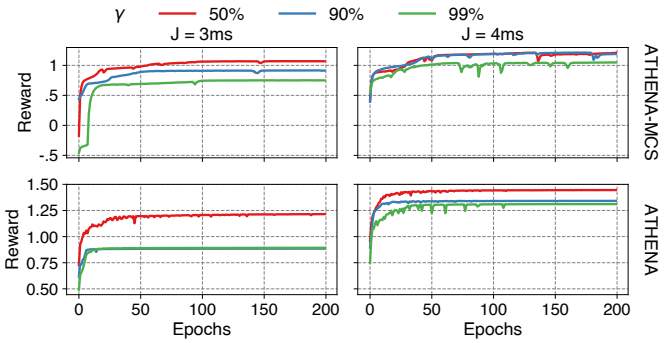

Fig. 5. Actor's objective across training epochs.

A. `ATHENA-ML` convergence

In Fig. 5, we depict the actor's objective, *i.e.*, the critic's average reward prediction.

We 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. We set $r_{thres} = 90\%$, since 3GPP defines 10% maximum BLock Error Rate (BLER), and penalty factor $K = 1$. We notice that 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 γ 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. Conversely, lower γ 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. We observe that `ATHENA-ML` performs better in terms of the average actor's objective than `ATHENA-MCS`, which we also validate in Sec. VI-B.

Through the rest of the evaluation section, unless otherwise stated, we will refer to the $J = 4$ ms scenario, which is typically used in the eMBB case, and $\gamma = 99\%$ to ensure high reliability.

B. `ATHENA-ML` performance

We now compare `ATHENA-ML` against `Baseline`. For PRB selection, it implements a Round Robin Policy, assigning

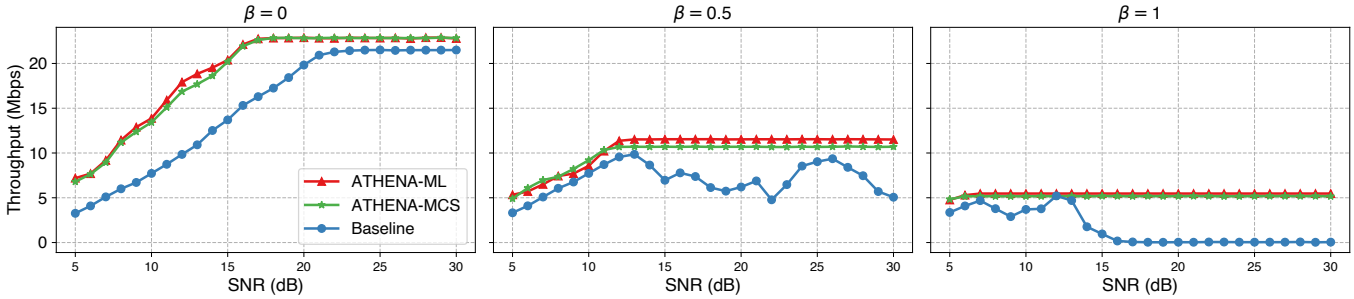

Fig. 6. Throuhgput served by a gNB running ATHENA-ML, ATHENA-MCS and Baseline algorithm, for different congestions β and SNR between $[5, 30]$ dB.

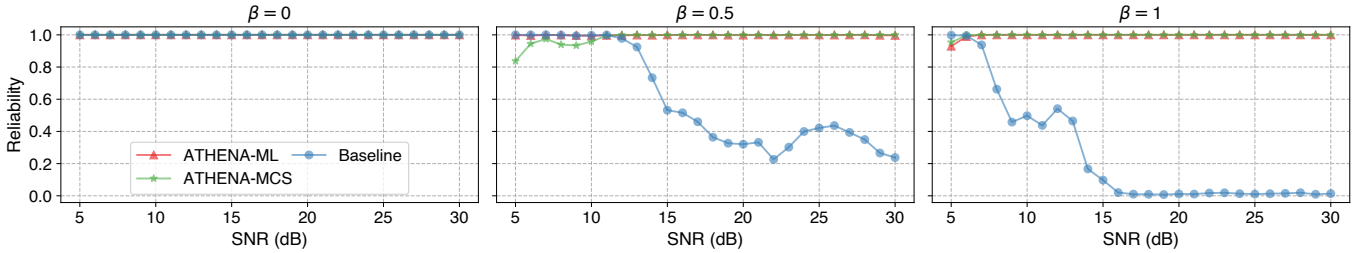

Fig. 7. Reliability by a gNB running ATHENA-ML, ATHENA-MCS and Baseline algorithm, for different congestions β and SNR between $[5, 30]$ dB.

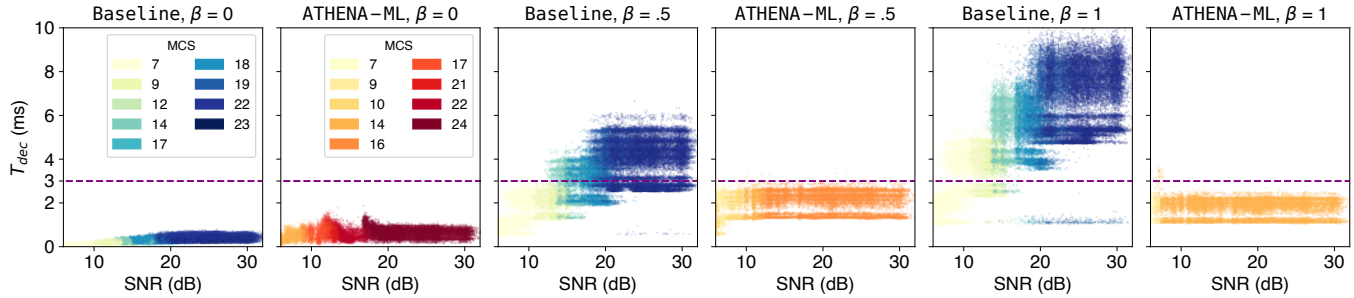

Fig. 8. The T_{dec} distribution and the selected MCS, for ATHENA-ML and Baseline algorithm, with different congestion β.

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 [56] 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. We 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.

We ran our experiments using TTI of 1 ms and 10 MHz of bandwidth, which gives 50 PRBs in the uplink out of which the $N = 45$ are only available for uplink data transmission. We have also disabled HARQ retransmissions in order to compare the success of our solution during the first transmission of the frame, *i.e.*, using a single redundancy version. We adjust 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. 6, we plot in solid lines the average throughput (in Mbps) achieved by ATHENA-ML, ATHENA-MCS and Baseline at each SNR level. A TB is considered successfully decoded when T_{dec} is below 3 ms and has a successful CRC check. When $\beta = 0$, where either controller suffers losses from deadline violations, Baseline substantially underperforms both ATHENA-ML and ATHENA-MCS, in terms of average 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).

We observe similar behavior in higher $\beta = .5$, where the congestion factor has an important impact on the decoding time. Baseline's throughput drops due to low reliability, as we depict in Fig. 7, 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.

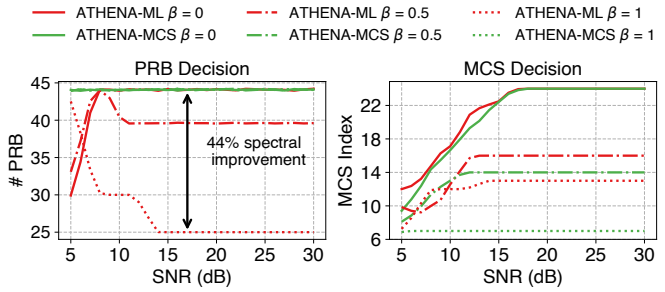

Fig. 9. MCS and PRB decisions of the two versions ATHENA-ML and ATHENA-MCS.

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

In Fig. 8, we compare the T_{dec} yielded by the two approaches. While in $\beta = 0$, all T_{dec} are below 3 ms, we observe that variability is much higher in ATHENA-ML, which is explained by higher MCS selection. Again, with higher β, a non-negligible part of the frames misses the deadline. ATHENA-ML, instead, lowers down 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, as we discuss next.

C. Advantages of the joint MCS-PRB management

We now compare the two alternate versions of ATHENA-ML and ATHENA-MCS. In Fig. 9, we plot 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 we observe in Fig. 6, both versions manage to achieve optimal levels of reliability and equal performance, with ATHENA-ML slightly overseeding by 500 Kbps in medium congestion factors, transmitting at lower PRBs and increasing the MCS. The major advantage by 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 TTI, 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. We note that both the original ATHENA-ML algorithm and ATHENA-MCS are both based on the same original architecture and should be considered variants and not alternative solutions.

D. Multi-user scenario

As explained in Sec. V-C, we resort to DT in order to evaluate the performance of our solution in a multi-user setting. We consider 3 users in the UL with increasing mean SNR and standard deviation of 3 dB. In each TTI, 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 DT 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.

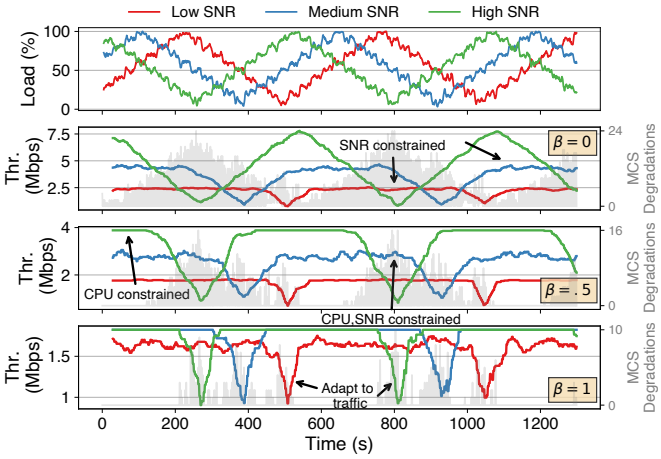

Fig. 10. Throughput for different congestion factor β and user load.

To capture the capability of ATHENA-ML to adapt to different UL traffic shapes, we modify the load as we depict in the first row of Fig. 10 in a period of 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, we show how the throughput is modified for each user. We see that the throughput, achieved by ATHENA-ML, follows the pattern of the traffic load which incurs due to MCS degradations, as we described in Sec. V-B5, and are shown in the shaded gray area.

In Fig. 11, we repeat the same experiment but for full traffic load (100%). We depict with bars the mean throughput and with error lines the jitter of each user for variable β values. We observe that in low β 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 no account since the congestion impact oversedes. In the same plot, we depict the cell utilization, defined as the total throughput of the users divided by the throughput at $\beta = 0$, which denotes the maximum achievable throughput for these channel conditions. We observe a gradual drop of cell utilization in low congestion factors, reaching 60% in medium β, before falling to below 40% in high β, showcasing its impact in the model's predictions.

VII. Explainability through Machine Reasoning

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

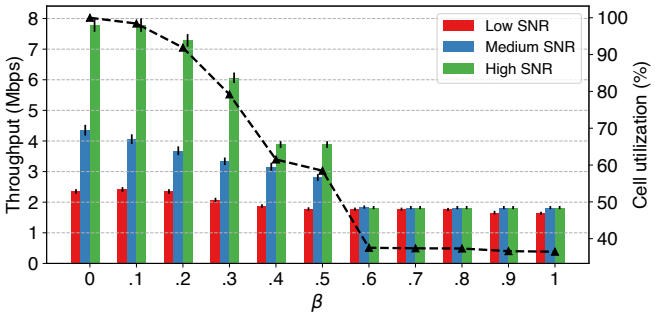

Fig. 11. Throughput for different congestion factor β and cell utilization.

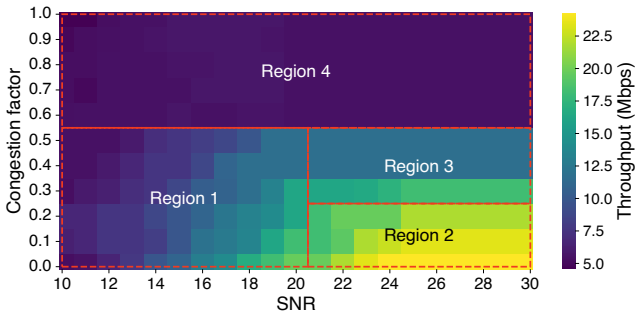

Fig. 12. ATHENA-ML throughput and the regions of the distilled decision tree.

The model has to provide insights into its internal functionality and decision process, which we call *explanations*. These explanations are categorized into three different classes depending on the question that they seek to answer [13]; the ***attributive***, the ***contrastive***, and the ***actionable*** explanations. In the study we conducted, we investigated methods for all three categories and we adjusted them accordingly to our NN-based actor-critic architecture.

A. 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 [57] and SHAP [58], which are model-agnostic explanators.

Model distillation is another method that can be used to produce explanations. The key concept behind is to distill a *black-box* model (such as a neural network) into an inherently better *explainable* surrogate model. This class of surrogate models typically comprises swallow decision trees and linear models, whose parameters (splitting criteria and variable coefficients respectively) are generally admitted to be better understandable. The distillation process consists of two models, namely the *teacher* $\mathcal{T}$ and the *student* $\mathcal{S}$. $\mathcal{T}$ is the black-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. 3, consists of two neural networks; the actor's μ_θ, which provides an approximate solution $\hat{\alpha}$, and critic's Q_ϕ, which assists to refine it to α. We distill the actor's model to a regression tree, which is a decision tree that holds linear models in its leaves (instead of constant approximators). We used *response-based knowledge distillation* [59], where $\mathcal{S}$ 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. In order to train the tree $\mathcal{S}$, we pass these outputs 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 our case is the L_2-distance, capturing the difference in yielded throughput between $\mathcal{T}$ and $\mathcal{S}$. In fact, we use the teacher's predictions as target values and we train the decision tree $\mathcal{S}$ in a supervised way, using the distillation loss as split criteria. We trained with maximum tree depth 3, in order to make it easier to interpret.

In Fig. 12, we depict the achieved throughput of the agent for combinations of the contextual features, and with the red dashed-edged rectangles we paint the borders of the leaves of the trained decision tree. We interpret the splitting thresholds in order to understand 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 β, 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.

B. Contrastive Explanations

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

To adapt this question to our case study, we 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* $\alpha' = (n', m')$*?* To answer this question, we leverage the architecture and resort to the objective function of the actor in Eq. 2, *i.e.*, to maximize the reward prediction of the critic. 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 Fig. 13, for the contextual state $\beta = .6$ and $\bar{c} = 20$ dB, we show the critic's reward prediction for all possible actions. From the figure we can see that ATHENA-ML actor learns one of the actions that yield the highest predicted reward, effectively approximating in its

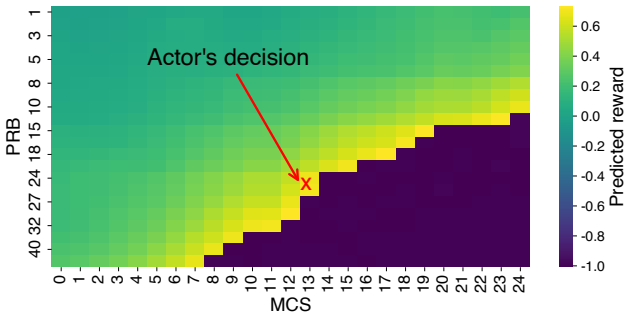

Fig. 13. Critic's reward prediction for SNR 20 dB and congestion factor $\beta = .6$.

internal variables where the negative reward area (*i.e.*, a frame not decoded in time) is hit.[3]

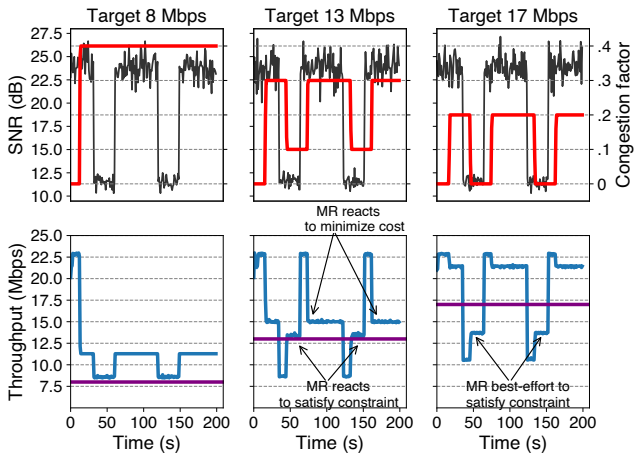

Fig. 14. Congestion factor re-orchestration by ATHENA−MR to satisfy KPI.

C. 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. 1. There may exist multiple counterfactuals that answer the question above, but we only consider the counterfactual that requires the smallest possible change [31].

For this analysis, we hence 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 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 β, since the user's channel conditions depend on many exogenous factors related to the UE characteristics. Because configuring the minimum β is the obvious answer, we introduce a sustainability clause in the cost function $C(\beta) : \mathcal{R} \rightarrow \mathcal{R}$ of operating at a certain β, 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, we only draw a simple requirement constraining $C(\beta)$ to be a monotonically decreasing function. This is motivated by real systems, as high β 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 [31]:

$$\begin{aligned} \min_{\beta \in \mathcal{B}} \quad & C(\beta) \\ \text{s.t.} \quad & thr^{\beta}_{\texttt{ATHENA-ML}} \geq thr_{target} \end{aligned} \tag{6}$$

where the constraint assures that the throughput achieved by ATHENA−ML $thr^{\beta}_{\texttt{ATHENA-ML}}$ at congestion factor β satisfies the target.

The MR block of ATHENA, namely ATHENA−MR, analyzes the different input states, over a time window, in order 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.

To solve the optimization, we devised a simple heuristic. A replica of ATHENA−ML agent exists in the Non-RT RIC. We act reactively and consider the user's minimum SNR over the last window, *i.e.*, the worst-seen channel condition, iterate over the possible $\beta_i \in \mathcal{B}$, infer ATHENA−ML's achievable throughput and retrieve the minimum β that satisfies the constraint in Eq. 6.

To showcase the capability of the ATHENA−MR to re-orchestrate the congestion factor, we consider a UE at full buffer demand and we modify the gain G of the channel 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. In order 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), we set the feasible set of possible actions equal to $\mathcal{B} = \{0, .1, .2, \ldots, 1\}$.

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. We consider 3 different scenarios with 3 desirable throughputs; 8, 13, and 17 Mbps. In the top row of Fig. 14, we show in black

[3]It is important to clear out that the agent outputs directly the best action and we do not infer the critic for all the possible actions, which would be intractable.

the SNR per PRB and in red the re-orchestrated β. In the second row, we show 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. We 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, β 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.

In Fig. 4, we see 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.

VIII. Conclusion

In this paper, we presented ATHENA, a machine learning framework for the radio resource management in vRAN systems, that optimizes the achieved throughput according to the status of the underlying cloud infrastructure. We discuss ATHENA design, implementation, and alignment with the most relevant network standards. Moreover, we propose the concept of *actionable* Machine Reasoning, which takes further decisions based on the decisions taken by the Machine Learning algorithm. We show how ATHENA outperforms the vanilla radio resource controller available in the baseline implementation of a gNB and discuss the explainability of the proposed system, especially in terms of actionable explanations.

Acknowledgement

The work of University Carlos III of Madrid has been funded by European Union's Horizon-JU-SNS-2022 Research and Innovation Programme Project TrialsNet (Grant Agreement No. 101095871) and H2020 DAEMON (Grant Agreement No. 101017109). This work is also partially supported by the Spanish Ministry of Economic Affairs and Digital Transformation and the European Union-NextGenerationEU through the UNICO 5G I+D 6G-CLARION project.

References

[1] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Study on Artificial Intelligence (AI)/Machine Learning (ML) for NR air interface." 3GPP TR 38.843 Release 18, May 2022.

[2] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Study on Enablers for Network Automation for 5G System (5GS)." 3GPP TR 23.700-81 Release 18, May 2022.

[3] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Management and Orchestration; Study on AI/ML management." 3GPP TR 28.908 Release 18, July 2022.

[4] ENI, "Experiential Networked Intelligence (ENI): System Architecture." ENI Group Specification, Dec 2021.

[5] ZSM, "Enablers for AI-based Network and Service Automation." ZSM012 Group Specification, July 2022.

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

[7] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA) and NR; Study on enhancement for Data Collection for NR and EN-DC." 3GPP TR 37.817 version 17.0.0 Release 17, Dec. 2022.

[8] 3GPP, "3rd Generation Partnership Project; Architecture enhancements for 5G System (5GS) to support network data analytics services." 3GPP TS 23.288 version 17.0.0 Release 17, Dec. 2022.

[9] 3GPP, "3rd Generation Partnership Project; A Management and orchestration; Management Data Analytics (MDA)." 3GPP TS 28.104 version 17.0.0 Release 17, Dec. 2022.

[10] M. Gramaglia, M. Kajo, C. Mannweiler, O. Bulakci, and Q. Wei, "A unified service-based capability exposure framework for closed-loop network automation," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 11, p. e4598, 2022.

[11] 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.

[12] W. Samek, T. Wiegand, and K. Müller, "Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models," *CoRR*, vol. abs/1708.08296, 2017.

[13] 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.

[14] 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.

[15] 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*, MobiCom '21, (New York, NY, USA), p. 749–761, Association for Computing Machinery, 2021.

[16] srsRAN, "Open Source SDR 4G/5G Software Suite from Software Radio Systems (SRS)." GitHub, Aug. 1, 2022 [Online].

[17] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Study on 5G System Support for AI/ML-based Services." 3GPP TR 23.700-80 Release 18, May 2022.

[18] ORAN, "ORAN Working Group 2; AI/ML Workflow Description and Requirements." ORAN Technical Report, October 2020.

[19] 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*, pp. 1–7, IEEE, 2017.

[20] 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, 2018.

[21] 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)*, pp. 1–6, IEEE, 2019.

[22] F. Al-Tam, N. Correia, and J. Rodriguez, "Learn to Schedule (LEASCH): A Deep Reinforcement Learning Approach for Radio Resource Scheduling in the 5G MAC layer," *IEEE Access*, vol. 8, pp. 108088–108101, 2020.

[23] J. Bartelt, P. Rost, D. Wubben, J. Lessmann, B. Melis, and G. Fettweis, "Fronthaul and Backhaul Requirements of Flexibly Centralized Radio Access Networks," *IEEE Wireless Communications*, vol. 22, no. 5, pp. 105–111, 2015.

[24] 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*, pp. 125–136, 2012.

[25] 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.

[26] 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.

[27] J. A. Ayala-Romero, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, A. Banchs, and J. J. Alcaraz, "VrAIn: A Deep Learning Approach Tailoring Computing and Radio Resources in Virtualized RANs," in *The 25th Annual International Conference on Mobile Computing and Networking*, MobiCom '19, (New York, NY, USA), Association for Computing Machinery, 2019.

[28] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Muller, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019.

[29] J. Philips, C. Hahn, P. Fontana, D. Broniatowski, and M. Przybocki, "Four Principles of Explainable Arificial Intelligence," *National Institute of Standards and Technology (NIST) U.S. Department of Commerce*, 2020.

[30] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu, "Definitions, Methods, and Applications in Interpretable Machine Learning," *Proceedings of the National Academy of Sciences*, vol. 116, no. 44, pp. 22071–22080, 2019.

[31] 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.

[32] A. Banchs, M. Fiore, A. Garcia-Saavedra, and M. Gramaglia, "Network Intelligence in 6G: Challenges and Opportunities," in *Proceedings of the 16th ACM Workshop on Mobility in the Evolving Internet Architecture*, MobiArch '21, (New York, NY, USA), p. 7–12, Association for Computing Machinery, 2021.

[33] 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, pp. 52–58, nov 2022.

[34] 3GPP, "3rd Generation Partnership Project; 5G; NR; Overall description; Stage-2." 3GPP TS 38.300 Release 18, May 2022.

[35] 3GPP, "3rd Generation Partnership Project; Technical Specification Group Radio Access Network; 5G; NR; Physical channels and modulation." 3GPP TS 38.211 Release 17, 2023.

[36] 3GPP, "3rd Generation Partnership Project; 5G; NR; Physical layer procedures for data." 3GPP TS 38.214 Release 17, Jan. 2023.

[37] 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.

[38] 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)*, pp. 1–6, 2019.

[39] 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*, CoNEXT '20, (New York, NY, USA), p. 232–244, Association for Computing Machinery, 2020.

[40] 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.

[41] 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*, SIGCOMM '20, (New York, NY, USA), p. 270–282, Association for Computing Machinery, 2020.

[42] Y. Sun and J. R. Cavallaro, "A Flexible LDPC/Turbo Decoder Architecture," *Journal of Signal Processing Systems*, vol. 64, no. 1, pp. 1–16, 2011.

[43] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "SrsLTE: An Open-Source Platform for LTE Evolution and Experimentation," in *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, WiNTECH '16, (New York, NY, USA), p. 25–32, Association for Computing Machinery, 2016.

[44] 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.

[45] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[46] 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," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2016.

[47] G. Dulac-Arnold, R. Evans, P. Sunehag, and B. Coppin, "Reinforcement Learning in Large Discrete Action Spaces," *CoRR*, vol. abs/1512.07679, 2015.

[48] M. Series, "Minimum requirements related to technical performance for IMT-2020 radio interface(s) Report ITU-R M.2410-0," 2017.

[49] V. F. Monteiro, M. Ericson, and F. R. P. Cavalcanti, "Fast-RAT Scheduling in a 5G Multi-RAT Scenario," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 79–85, 2017.

[50] 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)*, p. 518–523, IEEE Press, 2022.

[51] L. Bonati, P. Johari, M. Polese, *et al.*, "Colosseum: Large-Scale Wireless Experimentation Through Hardware-in-the-Loop Network Emulation," in *Proc. of IEEE Intl. Symp. on Dynamic Spectrum Access Networks (DySPAN)*, (Virtual Conference), December 2021.

[52] 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.

[53] N. Apostolakis, L. E. Chatzieleftheriou, D. Bega, M. Gramaglia, and A. Banchs, "Digital Twins for Next-Generation Mobile Networks: Applications and Solutions," *IEEE Communications Magazine*, pp. 1–7, 2023.

[54] 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.

[55] B. Rabenstein and J. Volz, "Prometheus: A Next-Generation Monitoring System (Talk)," (Dublin), USENIX Association, May 2015.

[56] 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.

[57] 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*, KDD '16, (New York, NY, USA), p. 1135–1144, Association for Computing Machinery, 2016.

[58] S. M. Lundberg and S. Lee, "A unified approach to interpreting model predictions," *CoRR*, vol. abs/1705.07874, 2017.

[59] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, pp. 1789–1819, mar 2021.

Nikolaos Apostolakis is a Ph.D. student, affiliated with the University Carlos III of Madrid (UC3M) and the IMDEA Networks Institute. He holds an M.Sc. (2017) in electrical and computer engineering and a Master's degree (2021) in NFV/SDN for 5G Networks. His interests include virtualized mobile networks and machine learning.

Marco Gramaglia is a professor at UC3M, where he got his Ph.D. in 2012. Dr. Gramaglia's research interests are machine learning applied to 6G Networks, data privacy, and cybersecurity. He has been involved in several projects funded by the European Commission through the 5GPPP and SNS and he has authored more than 80 peer-reviewed publications.

Livia Elena Chatzieleftheriou is a post-doctoral researcher at the IMDEA Networks Insitute and UC3M, and a part-time lecturer with UC3M. She holds an M.Sc. (2015) in applied mathematics, and a Ph.D. (2022) in Resource allocation, content recommendation and online learning mechanisms for mobile edge networks. Her current research interests are in online learning and explainable AI for next-generation mobile networks.

Tejas Subramanya received the M.S. degree in radio communications from Aalto University, Finland, and the Ph.D. degree in cognitive network management and orchestration from the University of Trento, Italy. He is a Senior Research Engineer with Nokia Standards, Munich, Germany. He has over eight years of experience in mobile networks related research and development in different roles in India, Finland, Italy, and Germany. He is a coauthor of several articles and papers on network management automation for next-generation mobile networks. His current research interests include cognitive management of future mobile networks and applying machine learning to network automation use cases.

Albert Banchs has a double affiliation as a professor at the University Carlos III of Madrid and deputy director of IMDEA Networks Institute. He is an author of more than 100 publications, has been the Principal Investigator of 9 European Projects, and has served on many TPCs and Editorial Boards. He received his M.Sc. and Ph.D. degrees from the Polytechnic University of Catalonia (UPC) in 1997 and 2002.

Henning Sanneck heads the "Network Automation" Research Department in the Standards unit of Nokia Strategy & Technology, Munich, Germany. He received his Dr.-Ing. (PhD) degree in Electrical Engineering from the Technical University of Berlin in 2000. His current research interests are in (Beyond) 5G Network Management and Orchestration, in particular configuration, healing and the operation of Cognitive Functions in virtualized, sliced radio access networks (across public and private deployment scenarios). Henning has published 80 papers and has 30 patents granted or published.