Reinforcement Learning for Inter-Operator Sharing in Open-RAN

Mahdi Sharara, Sahar Hoteit, Véronique Vèque

▶ **To cite this version:**

Mahdi Sharara, Sahar Hoteit, Véronique Vèque. Reinforcement Learning for Inter-Operator Sharing in Open-RAN. IEEE INFOCOM 2024 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), May 2024, Vancouver, Canada. hal-04504786

HAL Id: hal-04504786
https://hal.science/hal-04504786

Submitted on 14 Mar 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Reinforcement Learning for Inter-Operator Sharing in Open-RAN

Mahdi Sharara*, Sahar Hoteit†, and Véronique Vèque†
*Orange, Innovation Division, Paris, France
†Université Paris Saclay-CNRS-CentraleSupélec, *Laboratoire des Signaux et Systèmes, 91190, Gif-sur-Yvette, France*
Emails: mahdi.sharara@orange.com, {sahar.hoteit, veronique.veque}@universite-paris-saclay.fr

***Abstract*—Towards Beyond 5G and 6G, Open Radio Access Network (Open-RAN) is a recent RAN architecture that promotes the decoupling of RAN components, virtualization, open interfaces, and the use of machine learning-based intelligent models. Operators can benefit from this architecture to optimize the network performance and reduce deployment and operation costs. Open-RAN paves the way for RAN-sharing, where multiple operators can share the same infrastructure instead of deploying their own. In this paper, we model the problem of allocating radio and computing resources to multiple operators with different services as an Integer Linear Programming (ILP) problem aiming to satisfy users' demands. Due to the high complexity of solving an ILP problem, we develop a policy-gradient-based Reinforcement Learning (RL) model that aims to dynamically allocate resources to operators. The simulation results demonstrate the higher efficiency of our RAN-sharing RL model as it improves the radio and CPU resource utilization compared to No-Sharing models that deploy double the amount of provisioned resources, as each operator has its own infrastructure, with its own base stations and computing resources. In the considered scenario, RL demonstrates up to 19.5% more RBs utilization and 27.4% more CPU utilization. This highlights the ability of the RL model to reduce operational and deployment costs. Additionally, the RL model outperforms static RAN-sharing algorithms thanks to its dynamic adaptation to operators' varying traffic. Precisely, it scores up to 12.3% and 17.6% more RBs and CPU utilization, respectively.**

I. Introduction

As the demand for mobile data continues to multiply, the network architecture continues to evolve to accommodate various services and adapt to customers' needs and demands. Open Radio Access Network (Open-RAN) is a new paradigm that promotes the openness and virtualization of different Radio Access Network (RAN) components and interfaces. The Open-RAN architecture supports having components from multiple vendors and ensures that the interfaces connecting these components are open. Such openness is needed to achieve interoperability. It also encourages competition among vendors, reducing capital and operational expenditure [1]. In this architecture, a traditional base station is decoupled into an Open Radio Unit (O-RU), an Open Distributed Unit (O-DU), and an Open Central Unit (O-CU) [2]. The O-RU is responsible for Radio Frequency functions and lower physical (low-PHY) layer functions. The O-CU is responsible for functions of the higher layers, such as the Packet Data Convergence Protocol (PDCP) and Radio Resource Control (RRC) layers. The O-DU is responsible for the High Physical layer (High-PHY), Multiple Access Control (MAC), and Radio Link Control (RLC) layers [3]. As the network evolves toward 6G, Open-RAN should be able to provide improved performance and reduced costs from operators' perspective.

One of the main goals of operators is to optimize profits. Operators aim to reduce deployment and operational costs. Hence, the concept of sharing RAN resources among multiple operators raises as a candidate to reduce costs. For example, in some rural areas that rarely have a high demand, the utilization of radio and computing resources is likely to be low. In legacy RAN architecture, operators used to deploy non-shared physical or virtual resources in these areas or to rent them from an infrastructure provider. Instead of deploying (or renting) resources that are on average underutilized, with O-RAN architecture, it would be possible to deploy shared infrastructure to improve the utilization of these resources. Better utilization indicates more efficient usage of resources as fewer resources are deployed, leading to less deployment and operational costs. Additionally, RAN sharing could be implemented as a backup plan in case an operator suddenly gets overloaded or has a temporary system failure.

Enabling RAN sharing requires coordination between network components in addition to intelligent and central control. Open-RAN standardizes the near-Real-Time Radio Intelligent Controller (near-RT-RIC) and the non-Real-Time Radio Intelligent Controller (non-RT-RIC) [4]. These RICs provide closed-loop control to manage the different network components. Additionally, they allow data-driven algorithms to exploit the massively available data to optimize the network performance. Precisely, the RICs permit training, deployment, and optimization of different Machine Learning-based algorithms. To reap the benefits of RAN sharing, it is necessary to devise algorithms that efficiently distribute the resources to the involved operators. Such algorithms could be based on machine learning, which can solve various problems in telecommunication [3].

In this paper, we consider the problem of inter-operator RAN sharing, considering a scenario in which both the spectrum and the infrastructure are shared. We model an Integer Linear Programming (ILP) problem that allocates radio and computing resources to different users from multiple operators, aiming to satisfy users' demands from all operators. Due to the high complexity of solving an ILP problem, we propose a Reinforcement Learning (RL)-based algorithm to dynamically solve the allocation problem. We compare the RL problem to other benchmarking algorithms; Equal-Sharing, Proportional-Sharing, and No-Sharing. Precisely, we evaluate the ability of the RL-based model to better utilize the radio and computing resources and to serve more demands.

The rest of the paper is organized as follows: Some related works are surveyed in Section II. The ILP problem is formulated in Section III, and the RL model is presented in Section IV. The simulation results are discussed in Section V, and our work is concluded in Section VI.

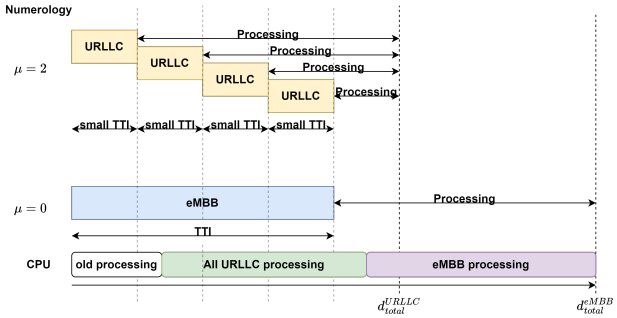

Fig. 1: Higher numerology allows for shorter TTI, which we call small TTI, leaving more time window for baseband processing

II. Related Work

Some research papers have considered partial RAN sharing in Cloud-RAN, where only the spectrum is shared. We recall that Cloud RAN is a RAN architecture that centralizes and virtualizes RAN functions. This architecture has been boosted by Open-RAN with the additional support for multi-vendors, openness, and standardized Radio Intelligent Controllers. Authors in [5] consider beamforming control when users of operators are connected to multiple Radio Units. It aims to minimize the total power consumption, including the fronthaul, the RRH, and the backhaul. They consider cooperation between operators: Part of the spectrum is private while the other is shared. They devise different algorithms and study the power consumption considering different bandwidth-sharing ratios among operators. Authors in [6] formulate a Mixed Integer Problem to share and allocate resources to operators with the goal of optimizing energy efficiency. They solve the problem through Lagrange dual composition. In [7], authors adopt the Open-RAN architecture. They propose to use an algorithm based on Deep-Deterministic Policy Gradient along with federated learning by virtual operators to learn the optimal allocation to slices focusing on Service Level Agreement (SLA) satisfaction. Authors in [8] propose a platform for automated zero-touch network and spectrum sharing in Open-RAN. They model the problem using Quadratically Constrained Quadratic Programming. They note an improvement in per-user average throughput.

Through the Radio Intelligent Controllers, the Open-RAN architecture provides native support for training and deploying Machine Learning algorithms. Different Machine learning techniques, including Reinforcement Learning, have been used to solve various problems in mobile networks. A policy gradient-based algorithm with dynamic input size was used in [9] to schedule users dynamically. The used model can adapt to a dynamic number of users. Authors in [10] use a similar dynamic model to allocate computing resources to users while aiming to maximize fairness, whereas [11] uses a similar model to allocate computing resources while maximizing the operator profits.

Different from the aforementioned papers, and to the best of our knowledge, we are the first to propose a dynamic reinforcement learning algorithm that implements RAN sharing and that scales with the number of operators and services. Additionally, we are the first to tackle the problem from the perspective of improving resource utilization to reduce capital and operational expenditure.

III. Problem Formulation

We consider a system with a set of base stations $\mathcal{B}$. Mainly, the base station includes the antenna and the Open Radio Unit (O-RU) located physically at the cellular site, while the Open Distributed Unit (O-DU) and the Open Central Unit (O-CU) are run virtually in the O-Cloud. The spectrum and the CPU resources are shared by different operators belonging to set $\mathcal{O}$. The total radio Resource Blocks available for each base station b is denoted by N_b^{RB}. Operators have users from different service types. The set of services $\mathcal{V}$ includes enhanced Mobile Broadband (eMBB, $v = 0$) and Ultra Reliable Low Latency Communication (URLLC, $v = 1$). We denote by $\mathcal{U}_b^{o,v}$ the set of users of base station b belonging to operator o using service v. $\mathcal{N}$ is the set of integer numbers in the interval $[0, N_b^{RB}]$. In the Open Cloud (O-Cloud), we consider a set of CPU cores $\mathcal{C}$ to be allocated to operators to process users' frames. 5G introduces the concept of numerology, which defines the bandwidth of the subcarriers and the transmission duration. We consider numerology 0 for eMBB and numerology 2 for URLLC. Numerology 0 has a subcarrier spacing of 15KHz, while Numerology 2 has a subcarrier spacing of 60 KHz. In concrete, 1 RB in numerology 2 takes four times more space than numerology 0 in the frequency domain; however, it takes four times shorter transmission time. This allows for decreasing the transmission latency of URLLC frames.[1]

Suppose at the start of each Transmission Time Interval (TTI) (i.e., the time taken to transmit a full frame), eMBB frames are transmitted. The duration of this TTI is 1 ms. However, as URLLC uses numerology 2, the corresponding transmission duration is four times shorter than that of eMBB. We refer to it in this paper as small TTI that equals 0.25 ms. This allows four consecutive URLLC transmissions during 1 ms. Fig. 1 shows the impact of using the different numerologies on the transmission duration and the amount of time left for baseband processing. We suppose d^{eMBB} and d^{URLLC} are the time budgets of the sum of the transmission and baseband processing, for eMBB frames and URLLC frames respectively. We set d^{eMBB} to be equal to $2TTI = 2ms$ and d^{URLLC} to $1.25ms$. As the eMBB transmission counts for $1ms$, only $1ms$ is left for eMBB baseband processing. Meanwhile, for URLLC, the time available for processing depends on which small TTI they are transmitted. As the transmission duration of URLLC frames is shorter, they get additional time for baseband processing in case computing resources are available.

For simplicity, we suppose that at each TTI, the scheduling decisions for URLLC frames at the transmitter are simultaneously made for the upcoming 4 small TTIs. Considering uplink transmission, there will be four instants of arrivals of frames at the O-Cloud (i.e., the receiver) during each 1ms. These instants form the set $\mathcal{T} = \{1, 2, 3, 4\}$, where they are separated by 0.25ms. While URLLC frames can arrive at any of these instants, eMBB frames take 1ms for transmission, so they arrive at the last one ($t = 4$). Fig. 1 demonstrates how using different numerologies modifies the transmission time and hence, the time available for processing. Let $x_n^{b,o,v}$ be a binary variable that is equal to 1 if and only if operator $o \in \mathcal{O}$ is using n resource blocks of a base station $b \in \mathcal{B}$ for a service type $v \in \mathcal{V}$. Let $y_{n,c,t}^{b,o,v,u}$ be a binary variable that is equal to 1 if and only if user $u \in \mathcal{U}$ of service type $v \in \mathcal{V}$

[1]We note that multiplexing two numerologies creates Inter-Numerology-Interference and that using higher numerology could lead to higher delay spread, increasing errors due to multipath fading. In this work, we assume the operator takes the required measures to combat errors due to multipath fading, and we suppose that the resource blocks used for each service are adjacent (i.e., all eMBB RBs are adjacent, and all URLLC RBs are adjacent). To cope with the inter-numerology interference, we assume that a sufficient amount of the spectrum is available to be used as a guard band between the two numerologies.

and base station $b \in \mathcal{B}$ belonging to operator $o \in \mathcal{O}$ is using n resource blocks, CPU c for processing, and arrives at the O-Cloud at instant $t \in \mathcal{T}$. z_c^o is a binary integer that indicates that CPU c is assigned to operator o. Denote by $\rho_{dem}^{b,o,v,u}$ the demanded throughput of a user, $\rho_n^{b,o,v,u}$ the rate provided to a user when it uses n resource blocks, and $t_n^{b,o,v,u}$ the processing time of a user frame when it uses n resource blocks. As both the processing time and the number of resource blocks depend on the number of resource blocks and the Modulation and Coding scheme (MCS) index [12], we sample the MCS indexes of users from the distribution used in [13], [14]. This distribution provides the probability of having different MCS indexes based on real measurements. $\eta^{o,v}$ represents a priority weight for a service v of operator o while a user priority weight $\nu^{b,o,v,u}$ can be used to ensure fairness (i.e., lowering this values for users who have massively transmitted previously, and increasing it for those who haven't or transmitted less. The Integer Linear Programming model is shown here:

$$\max \sum_{b\in\mathcal{B}}\sum_{o\in\mathcal{O}}\sum_{v\in\mathcal{V}}\sum_{u\in\mathcal{U}_b^{o,v}}\sum_{n\in\mathcal{N}}\sum_{c\in\mathcal{C}}\sum_{t\in\mathcal{T}} \eta^{o,v}\nu^{b,o,v,u}\rho_n^{b,o,v,u}y_{n,c,t}^{b,o,v,u} \quad (1)$$

sub. to

$$x_n^{b,o,v} \in \{0,1\}, \forall b \in \mathcal{B}, o \in \mathcal{O}, v \in \mathcal{V}, n \in \mathcal{N} \quad (2)$$

$$y_{n,c,t}^{b,o,v,u} \in \{0,1\}, \forall b \in \mathcal{B}, o \in \mathcal{O}, v \in \mathcal{V}, u \in \mathcal{U}_b^{o,v}, n \in \mathcal{N}, c \in \mathcal{C}, t \in \mathcal{T} \quad (3)$$

$$z_c^o \in \{0,1\}, \forall o \in \mathcal{O}, c \in \mathcal{C} \quad (4)$$

$$\sum_{n\in\mathcal{N}} x_n^{b,o,v} \leq 1, \forall b \in \mathcal{B}, o \in \mathcal{O}, v \in \mathcal{V} \quad (5)$$

$$\sum_{o\in\mathcal{O}}\sum_{v\in\mathcal{V}}\sum_{n\in\mathcal{N}} n \cdot x_n^{b,o,v} \leq N_b^{RB}, \forall b \in \mathcal{B} \quad (6)$$

$$\sum_{n\in\mathcal{N}}\sum_{c\in\mathcal{C}}\sum_{t\in\mathcal{T}} y_{n,c,t}^{b,o,v,u} \leq 1, \forall b \in \mathcal{B}, o \in \mathcal{O}, v \in \mathcal{V}, u \in \mathcal{U}_b^{o,v} \quad (7)$$

$$\sum_{u\in\mathcal{U}_b^{o,v}}\sum_{n\in\mathcal{N}}\sum_{c\in\mathcal{C}} y_{n,c,t}^{b,o,v,u} \leq n \cdot x_n^{b,o,v}, \forall b \in \mathcal{B}, o \in \mathcal{O}, v = 0, t \in \mathcal{T} \quad (8)$$

$$\sum_{u\in\mathcal{U}_b^{o,v}}\sum_{n\in\mathcal{N}}\sum_{c\in\mathcal{C}} 4 \cdot y_{n,c,t}^{b,o,v,u} \leq n \cdot x_n^{b,o,v}, \forall b \in \mathcal{B}, o \in \mathcal{O}, v = 1, t \in \mathcal{T} \quad (9)$$

$$\sum_{b\in\mathcal{B}}\sum_{o\in\mathcal{O}}\sum_{u\in\mathcal{U}_b^{o,v}}\sum_{n\in\mathcal{N}}\sum_{c\in\mathcal{C}} y_{n,c,t}^{b,o,v,u} = 0, \forall t \in \{1,2,3\}, v = 0 \quad (10)$$

$$\sum_{b\in\mathcal{B}}\sum_{o\in\mathcal{O}}\sum_{u\in\mathcal{U}_b^{o,v}}\sum_{n\in\mathcal{N}}\sum_{t'\in\mathcal{T}'} y_{n,c,t'}^{b,o,v,u} \cdot t_n^{b,o,v,u} \leq d^{URLLC} - d_t - d_c^{old}, \forall c \in \mathcal{C}, v = 1, t \in \mathcal{T}, \mathcal{T}' = \{t' \in \mathcal{T}, t' \geq t\} \quad (11)$$

$$\sum_{b\in\mathcal{B}}\sum_{o\in\mathcal{O}}\sum_{u\in\mathcal{U}_b^{o,v}}\sum_{n\in\mathcal{N}}\sum_{t\in\mathcal{T}} y_{n,c,t}^{b,o,v,u} \cdot t_n^{b,o,v,u} \leq d^{URLLC} - d_c^{old}, \forall c \in \mathcal{C}, v = 1 \quad (12)$$

$$\sum_{b\in\mathcal{B}}\sum_{o\in\mathcal{O}}\sum_{u\in\mathcal{U}_b^{o,v}}\sum_{n\in\mathcal{N}} y_{n,c,t}^{b,o,v,u} \cdot t_n^{b,o,v,u} \leq d^{eMBB} - d_t, \forall c \in \mathcal{C}, v = 0, t = 4 \quad (13)$$

$$\sum_{b\in\mathcal{B}}\sum_{o\in\mathcal{O}}\sum_{v\in\mathcal{V}}\sum_{u\in\mathcal{U}_b^{o,v}}\sum_{n\in\mathcal{N}}\sum_{t\in\mathcal{T}} y_{n,c,t}^{b,o,v,u} \cdot t_n^{b,o,v,u} \leq d^{eMBB} - d_c^{old}, \forall c \in \mathcal{C} \quad (14)$$

$$\sum_{o\in\mathcal{O}} z_c^o \leq 1, \forall c \in \mathcal{C} \quad (15)$$

$$\sum_{b\in\mathcal{B}}\sum_{v\in\mathcal{V}}\sum_{u\in\mathcal{U}_b^{o,v}}\sum_{n\in\mathcal{N}}\sum_{t\in\mathcal{T}} y_{n,c,t}^{b,o,v,u} \leq z_c^o, \forall o \in \mathcal{O}, c \in \mathcal{C} \quad (16)$$

$$\sum_{n\in\mathcal{N}}\sum_{c\in\mathcal{C}}\sum_{t\in\mathcal{T}} y_{n,c,t}^{b,o,v,u} \cdot \rho_n^{b,o,v,u} \leq \rho_{dem}^{b,o,v,u}, \forall b \in \mathcal{B}, o \in \mathcal{O}, v \in \mathcal{V}, u \in \mathcal{U}_b^{o,v} \quad (17)$$

The objective (1) is used to maximize the total transmitted throughput, considering the priority weights given for operators, services, and users. Constraints (2), (3), (4) ensure the binary nature of the decision variables. (5) and (6) capture the total number of RBs used by operator o for service v at base station b. (7) ensures that each user gets a unique total number of resource blocks, its frame is processed on a unique CPU, and the frame is only transmitted once. Additionally, (8) ensures that the users of an operator of eMBB service at a specific base station can not use resource blocks more than what is assigned to their slice. We note that the RB in this ILP is defined according to numerology 0. To use numerology 2, 4 RBs are fused together to have a carrier spacing of 60 KHz instead of 15 kHz; this is ensured by constraint (9). (10) ensures that eMBB frames can not arrive at instants t=1,2,3 as their transmission takes 1 full TTI. Constraint (11) tracks the amount of processing budget available for URLLC alone after deducting the transmission and scheduling delay d_t and the processing from previous instants on the CPU d_c^{old}. Constraints (11) and (12) together ensure that the maximum of the URLLC transmission interval (0.25 ms) and d_c^{old} should be deducted of the available time budget of URLLC d^{URLLC} as it is not possible to process the frames during this interval. Constraint (13) tracks the time available for eMBB alone. If only eMBB is transmitted, there will be no overlap of processing, as the processing budget of an eMBB frame is equal to the TTI duration. On the other hand, (14) ensures that no overlapping of processing happens when URLLC frames use the CPU. Constraints (15) and (16) ensure that a CPU is assigned to one operator and only the users of this operator can use this CPU. Finally, (17) ensures that no user gets a rate higher than it demands.

In this part of the problem, we combine two problems: The first is the slicing among operators and services, and the second is the scheduling of users. While the latter is done on a small time scale (i.e., every $1ms$), the former is likely to be done on larger time scales. While Open-RAN supports Radio Intelligent Controllers (RIC) with $10ms$ and $1s$ time scales, the standardization of RealTime RIC (RT-RIC) is being considered, which would act on a time scale of $1ms$. Due to the high complexity of solving the ILP problem, we opt to use reinforcement learning to set upper-level policies. For scheduling users of each operator, we use the Proportional Fairness with priority for URLLC algorithm [15] to allocate RBs and CPU resources to users.

IV. Reinforcement Learning Model

In Reinforcement learning (RL), an agent learns by interacting with the environment. Precisely, the agent executes an action and receives an analysis of how good or bad the action is. The ultimate goal for an agent is to maximize its cumulative reward. RL could help solve complex problems, though suboptimally. To design the RL model, we define the state, action, and rewards for the RL agent. The state represents the current characteristics/observations of the agent within the environment. When the agent executes an action, the environment will provide it with a reward. Based on the rewards the agent receives, it will learn what actions will help it maximize the cumulative rewards, so it acts accordingly.

We consider a policy-gradient-based algorithm to allocate RBs to each service (i.e., eMBB and URLLC) for each operator and per each BS. In policy gradient algorithms, the agent learns the optimal probability distribution of actions that maximize its rewards. We represent each (Base Station, operator, service) tuple at a step i of the RL episode by $\delta_i^{b,o,v}$. An episode consists of multiple steps. At each step, an action (i.e., assigning a RB of a BS to an operator for

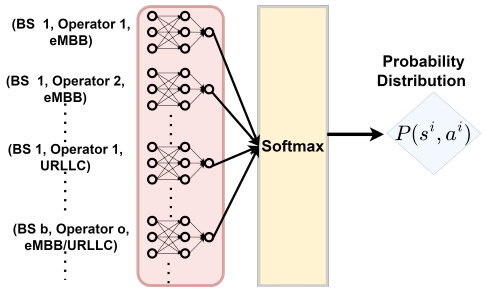

Fig. 2: Neural Network Architecture

a specific service) is taken. The episode is complete once all the resources are allocated. We use a flexible Neural Network, as in [9], [10], [11], that scales with the number of BSs, Operators, and services. The architecture is shown in Fig. 2. Each $\delta_i^{b,o,v}$ is fed into a common neural network to yield a value. The outputted values from all $\delta_i^{b,o,v}$ are fed into a softmax function producing a probability distribution. Thus, the agent selects a tuple according to this probability distribution. At each step, a tuple is selected, assigning an RB to an operator for a specific service and a specific base station. For simplicity and as we are considering a scenario with low load, the RL agent does not allocate CPUs to operators. When RAN sharing is enabled, we assume that operators share all the CPUs. As the RL deals with the upper-level decisions, the proportional fairness algorithm is used to allocate radio and CPU resources to schedule users' frames every TTI. However, we could adapt our model to include a tuple (CPU, operator, service) to allocate a CPU to an operator, and the state representation would include a variable to represent whether it is an RB or a CPU. We leave such adaptation for future work.

Next, we present the Markov Decision Process (MDP) of the RL agent, and the RL algorithm.

A. State

One episode consists of multiple steps. At each step, one tuple will be selected. An episode finishes when the RBs of all base stations have been distributed. Suppose that the instantaneous total number of packets in the queue at a base station b from operator o and service v is represented by $P^{b,o,v}$. To represent the tuple $\delta_i^{b,o,v}$ (Base Station, operator, service), we consider the total number of users for service v and operator o at base station b, the total number of users from each operator at base station b, and the total number of users from each service at base station b. Additionally, we consider the total number of non-served packets for service v and operator o at base station b. A flag (i.e. v) for the service type is also used. Additionally, $\delta_i^{b,o,v}$ includes the total number of allocated RBs for this tuple, which could vary based on the decisions from previous steps. Recall that $\mathcal{U}_b^{o,v}$ is the set of users at base station b of operator o using service v. $\delta_i^{b,o,v}$ is represented by:

$$\delta_i^{b,o,v} = \left\{ |\mathcal{U}_b^{o,v}|, \sum_{v \in \mathcal{V}} |\mathcal{U}_b^{o,v}|, \sum_{o \in \mathcal{O}} |\mathcal{U}_b^{o,v}|, P^{b,o,v}, v \right\}$$

The concatenation of all $\delta_i^{b,o,v}$ form the state s_i. We note that once all the RBs of a BS have been allocated, the tuples corresponding to this base station are removed from the state representation.

B. Action

At each step i of an episode, the goal is to allocate an RB of BS b to an operator o and service v. This is achieved by selecting a tuple that is represented by $\delta_i^{b,o,v}$. The action a^i at step i is the tuple:

$$a_i = (b, o, v), b \in \mathcal{B}, o \in \mathcal{O}, v \in \mathcal{V}$$

C. Reward

At each state s_i, once an action is executed, a reward is generated by the environment. This helps the RL agent assess how good its action has been. Recalling that $P^{b,o,v}$ is the total number of packets in the queue from operator o and service v at base station b, let $P_{served}^{b,o,v}$ be the total number of the served packets. For an action $a_i = (b, o, v)$, the rewards is defined by

$$r^i(s_i, a_i) = \eta^{o,v} \times (\frac{P_{served}^{b,o,v}}{P^{b,o,v}})$$

where $\eta^{o,v}$ is a weight used to control the priority for a service or operator.

D. RL algorithm

The RL agent uses the algorithm known as REINFORCE with a baseline [16]. In the training phase, the weights θ of the Neural Network are initialized before starting any episode. Then $\delta_i^{b,o,v}$ and state s^i are updated at each step i. Then the agent executes action a^i, gets r^i, and moves to s^{i+1}. This will be repeated until the terminal state is reached. This is the case when there is no more RBs to allocate to any operator at all BS. The weights θ are updated according to this formula:

$$\theta \leftarrow \theta + \alpha \gamma^i \nabla log(P(s^i, a^i))$$

α is the learning rate, $P(s^i, a^i)$ is the probability value yielded by the NN, and γ^i is the discounted reward. γ^i is normalized by subtracting the mean of the rewards in an episode and dividing by the standard deviation of the rewards.

V. Simulation and Results

To measure the performance of the RL algorithm for RAN sharing, we consider a scenario with 5 base stations. Each base station has 100 RBs. In the case of RAN sharing, the 100 RBs of each BS are shared by the operators who together share 4 CPUs in the O-Cloud. In case there is no sharing, there will be a BS for each operator, each with 100 RB. The same applies to the CPUs where each operator will use 4 CPUs. To measure the required processing time for user frames, we use the model in [12]. In fact, we are targeting the case where we have a low-to-medium load. This creates an incentive for operators to deploy shared infrastructure to reduce costs. We suppose that we have two operators and two services (i.e., eMBB and URLLC). We assume the combined deadline for transmission and baseband processing for eMBB is $2ms$, whereas it is $1.25ms$ for URLLC.

We set the weight $\eta^{o,v}$ to 1 for all the operators and services. The arrivals of users to base stations per service and per operator follow a Poisson distribution, where we vary the rate from 0.6 to 1 user/ms. The packets arrival per each eMBB user is also a Poisson distribution with a rate that is randomly chosen for each user such that it is between 1 and 50 packets per second. For each URLLC user, this rate is between 1 and 10 packets per second. A user persists in the system for a random time that is uniform between 10 and 30 ms. The size of eMBB packets is set to 1500 bytes whereas it is 32 bytes for URLLC.

The RL algorithm is initially trained for 4000 TTI. After training, we use the trained RL agent to test two variations.

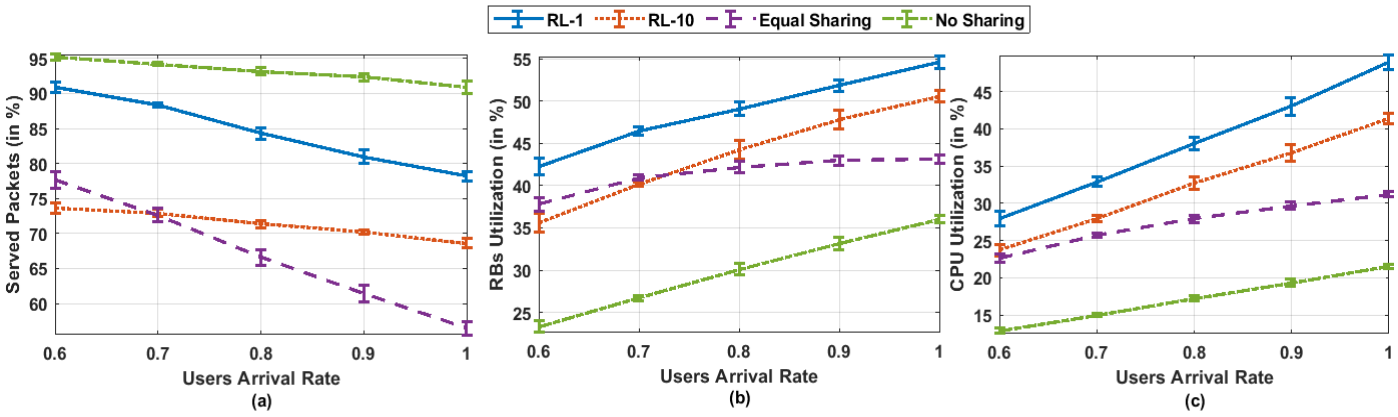

Fig. 3: (a) Percentage of served packets (b) RB Utilization (c) CPU Utilization as a function of users' arrival rate

RL-1 invokes the RL agent every $1ms$, so the system is reconfigured every $1ms$. In the second variation, *RL-10* algorithm is run every $10ms$ and reconfiguration of the system is done every $10ms$. The latter could be implemented in the near-RT-RIC. However, the former variation must be implemented in a Real-Time RIC that supports operations between 1 and 10 ms. The Real-Time RIC proposal could be considered in Open-RAN and is proposed in [17]. To compare the RL models, we consider three algorithms for benchmarking. The first one is called *Equal Sharing*, which does static sharing as in [18]. Assuming both operators have similar traffic patterns, each operator shares the same number of RBs, such that the 100 RBs of a BS are divided equally on each. Per operator, the 50 RBs are divided such that 38 RBs are allocated to eMBB with numerology 0. The remaining 12 RBs use numerology 2. Hence, every four RBs allocated to URLLC are grouped together to form one RB. We recall that in numerology 2, the transmission time (i.e., which we call small TTI) is 0.25 ms, and the subcarrier spacing is multiplied by 4 as 4 RBs get fused to form one URLLC RB. Hence, for every small TTI, we have 3 RBs (i.e., 12 initial ones divided by 4 fused RBs) for URLLC, but as we have 4 small TTIs during 1ms, the total available URLLC RBs within the windows of $1ms$ is equal to 12. Additionally, operators share 4 CPUs in the O-Cloud. Additionally, we consider the case where operators don't have the same arrival rate. So instead of *Equal sharing*, we use *Proportional Sharing* algorithm. In this algorithm, the RBs are not shared equally but proportionally according to the ratios of their arrival rates. On the other hand, we consider another policy *No Sharing* where each operator has its own resources (i.e., each operator has its own base station, each with 100 RBs and has 4 non-shared CPUs at the O-Cloud). We recall that the tested algorithms do not schedule users, as they all use the proportional fairness algorithm for low-level scheduling. To allow for the sharing of the CPUs, the proportional fairness combines all users from all base stations together. The proportional fairness ranks users based on the ratio of the achievable throughput if they get scheduled divided by the historical throughput. Users with the highest value are scheduled first under the condition that there are still enough computing resources to serve them. If the RBs of a specific (BS, service, operator) tuple are fully used, users belonging to such a tuple are excluded from the algorithm.

We repeat the simulation 10 times, and each run consists of 1000 TTI, and we plot the 95% confidence intervals. Figure 3 shows the percentage of served packets, the RBs utilization, and the CPU utilization as a function of the arrival rate of users. The served packets are the number of scheduled and processed packets normalized by the combined total number of packets in all the queues in the system. As expected, the *No Sharing* algorithm performs the best due to the over-provisioning of resources. This means the CAPEX and OPEX of the operators will be much higher compared to the case when sharing is permitted, as *No Sharing* requires doubling the resources. On the other hand, *RL-1* serves more packets compared to *Equal Sharing* as it benefits as it dynamically reallocates resources according to the instantaneous needs of each operator. Such demands fluctuate as packets arrive, and hence, resources can be moved from one operator to another or from one service to another to satisfy these demands. On the other hand, *RL-10* is not as good as *RL-1* since it takes more time to reconfigure the system. However, as the arrival rate increases, *RL-10* can perform better than *Equal Sharing* as the packets would arrive densely. So some users and packets will be massively served over a window of 10 ms, and the others have to wait 10 ms for the network to be reconfigured. Note that this could incur a high delay that would negatively affect delay-sensitive data.

Analyzing the RB utilization and CPU utilization in Fig. 3(b) and Fig. 3(c), the *No Sharing* has the lowest utilization. This results from the fact that each operator had to double its provisioning radio and processing resources. On the other hand, the RL algorithms have almost double the RB and CPU utilization of that of No Sharing, which means *RL-1* is using, on average, the same amount of resources as the *No Sharing* counterpart, given that *No sharing* has a doubled amount of resources. On the other hand, the RL algorithms, thanks to their dynamicity, are able to learn the instantaneous needs of the system. Thus, they better occupy RBs and CPUs. Next, we consider another case where operator 1 has a varying users' arrival rate while operator 2 has a fixed arrival rate of 1 user per ms. This allows us to test the case where one operator has a demand lower than the other operator. We recall that here we use *Proportional Sharing* algorithm instead of *Equal sharing*, as operators no longer have the same arrival rate. We also note that we did not retrain the model to capture this scenario, but we reused the model trained in the initial scenario. This is, in fact, a case of Transfer Learning where the knowledge of an agent in a specific scenario can be used in a close/similar but not identical scenario. Fig. 4 demonstrates the ability of the RL model to adapt to the dynamic demands and perform better than the baseline algorithm *Proportional*

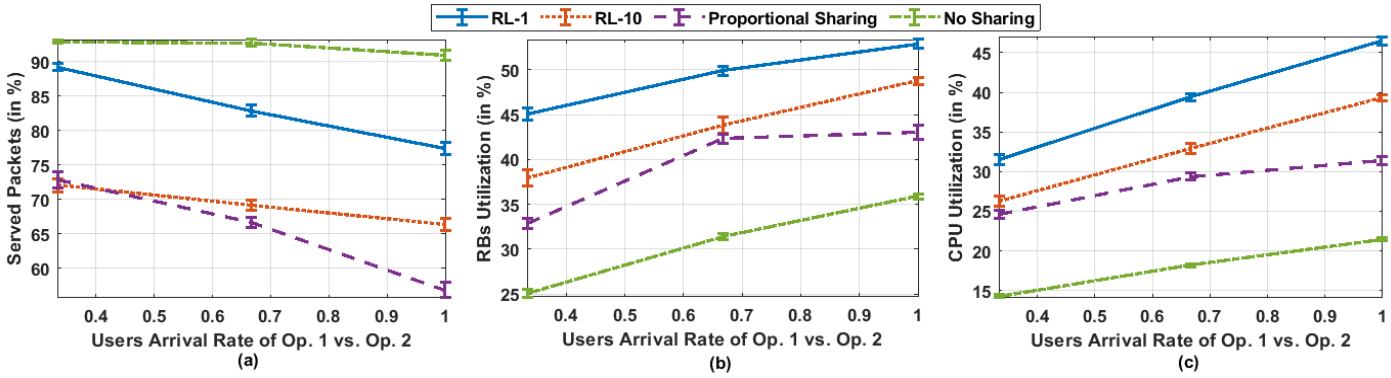

Fig. 4: Percentage of (a) Served Packets (b) RB Utilization (c) CPU Utilization as a function of the ratio users' arrival rate of operator 1 vs. operator 2

Sharing. Thanks to the dynamicity of the RL models, it will be able to respond to the non-frequent bursts of traffic arriving from the less-loaded operator 1.

These results demonstrate the efficiency of the sharing models to improve the radio and computing resources utilization compared to No-Sharing and thus allow for decreasing the deployment and operational costs.

VI. Conclusion

In this paper, we have considered the inter-operator sharing of radio and computing resources in Open-RAN. As operators aim to minimize the deployment and operational costs, sharing infrastructure presents a potential to decrease costs and improve profits. We have modeled the inter-operator sharing problem as an ILP problem that aims to share resources among operators while satisfying their users' demands. Due to the high complexity of solving an ILP problem, we developed an RL model to allocate resources among the different services of different operators. The results demonstrate the ability of the RL model to dynamically adapt to the varying traffic and dynamic demands of operators compared to static sharing models. Additionally, the RL model achieves higher radio and CPU resource utilization in comparison to the case of No sharing, where each operator has its own infrastructure. This demonstrates the ability of the RAN sharing enabled by the RL model to better utilize the resources, allowing operators to deploy fewer resources in total and share them. Hence, this decreases operators' capital and operational expenditure. In this work, the proposed RL module configures the allocation of RBs. For future work, we will modify the RL model so that it allocates both CPUs and RBs. Also, inspired by the advances in Natural Language Processing, we will consider using attention-based models such as transformers in the architecture of the RL model.

Acknowledgment

This work was funded by the ANR HEIDIS (https://heidis.roc.cnam.fr/; ANR-21-CE25-0019) project.

References

[1] A. K. U and G. Gundu Hallur, "Economic and Technical Implications of Implementation of OpenRAN by "RAKUTEN MOBILE"," in *2022 International Conference on Decision Aid Sciences and Applications (DASA)*, 2022, pp. 959–964.

[2] (2019) Cloud Architecture and Deployment Scenarios for O-RAN Virtualized RAN.

[3] M. Polese, L. Bonati, S. D'Oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, Interfaces, Algorithms, Security, and Research Challenges," *CoRR*, vol. abs/2202.01032, 2022. [Online]. Available: https://arxiv.org/abs/2202.01032

[4] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and Learning in O-RAN for Data-Driven NextG Cellular Networks," *IEEE Communications Magazine*, vol. 59, no. 10, 2021.

[5] J. Opadere, Q. Liu, T. Han, and N. Ansari, "Energy-Efficient Virtual Radio Access Networks for Multi-Operators Cooperative Cellular Networks," *IEEE Transactions on Green Communications and Networking*, 2019.

[6] S. Farhat, B. H. Hassan, and A. Ellatif Samhat, "Energy Efficient Resource Sharing in Multi-Operator Heterogeneous Cloud RAN," in *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2018, pp. 116–122.

[7] A. Abouaomar, A. Taik, A. Filali, and S. Cherkaoui, "Federated Deep Reinforcement Learning for Open RAN Slicing in 6G Networks," *IEEE Communications Magazine*, 2023.

[8] L. Bonati, M. Polese, S. D'Oro, S. Basagni, and T. Melodia, "NeutRAN: An open RAN neutral host architecture for zero-touch RAN and spectrum sharing," *IEEE Transactions on Mobile Computing*, pp. 1–14, 2023.

[9] J. S. Shekhawat, R. Agrawal, K. G. Shenoy, and R. Shashidhara, "A Reinforcement Learning Framework for QoS-Driven Radio Resource Scheduler," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–7.

[10] M. Sharara, T. Pamuklu, S. Hoteit, V. Vèque, and M. Erol-Kantarci, "Policy-Gradient-Based Reinforcement Learning for Computing Resources Allocation in O-RAN," in *2022 IEEE 11th International Conference on Cloud Networking (CloudNet)*, 2022, pp. 229–236.

[11] M. Sharara, S. Hoteit, and V. Vèque, "Reinforcement Learning based model for Maximizing Operator's Profit in Open-RAN," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, 2023, pp. 1–5.

[12] S. Khatibi, K. Shah, and M. Roshdi, "Modelling of Computational Resources for 5G RAN," in *2018 European Conference on Networks and Communications (EuCNC)*, 2018, pp. 1–5.

[13] M. Sharara, S. Hoteit, P. Brown, and V. Vèque, "Coordination between Radio and Computing Schedulers in Cloud-RAN," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2021.

[14] M. Sharara, S. Hoteit, P. Brown, and V. Vèque, "On coordinated scheduling of radio and computing resources in cloud-ran," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 2990–3003, 2023.

[15] M. Elsayed and M. Erol-Kantarci, "AI-Enabled Radio Resource Allocation in 5G for URLLC and eMBB Users," in *2019 IEEE 2nd 5G World Forum (5GWF)*, 2019, pp. 590–595.

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

[17] S. D'Oro, M. Polese, L. Bonati, H. Cheng, and T. Melodia, "dApps: Distributed Applications for Real-time Inference and Control in O-RAN," 2022. [Online]. Available: https://arxiv.org/abs/2203.02370

[18] M. Kassis, S. Costanzo, and M. Yassin, "Flexible multi-operator ran sharing: Experimentation and validation using open source 4G/5G prototype," in *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, 2021, pp. 205–210.