BLOCBIN: SLA-Compliant and Efficient RAN Slicing in Next-Generation Networks

Juliana Ojo, Guilherme Iecker Ricardo, Gentian Jakllari
IRIT/Toulouse INP, University of Toulouse, France - Email: name.surname@toulouse-inp.fr

***Abstract*—Efficient resource allocation in RAN Slicing poses significant challenges, particularly in meeting each slice's Service Level Agreements (SLAs) while ensuring isolation and customization. The choice of resource allocation method critically impacts the performance of users, slices, and the overall system. The inherent conflict between enforcing slice isolation and optimizing resource utilization makes the problem more difficult.**

We propose BLOCBIN, a channel-aware hierarchical downlink scheduling framework that ensures fairness relative to SLA across short- and long-term timescales while optimizing throughput. This framework effectively balances slice isolation with resource utilization efficiency and allows for customization. Simulation results show Blocbin improves SLA compliance by a factor of 1.5 compared to state-of-the-art solutions while maintaining the same spectral efficiency.

***Index Terms*—4g, 5g, RAN, network slicing, resource allocation, scheduling.**

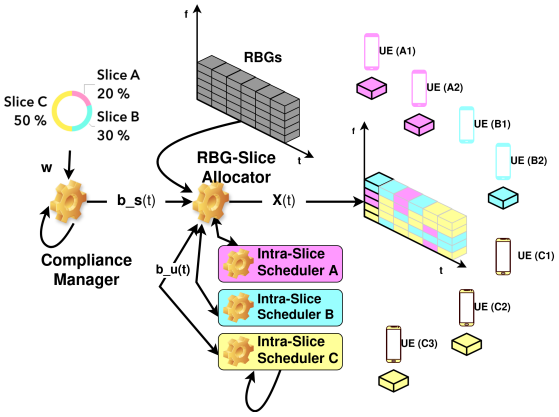


Fig. 1: SLA-compliant RAN slicing with BLOCBIN.

I. INTRODUCTION

Network slicing utilizes virtualization to establish multiple logical networks that operate concurrently on a shared physical infrastructure, thereby minimizing the reliance on specialized hardware. It is considered a key enabler for 5G and future network architectures, addressing the dual challenge of accommodating the ever-growing data demands of a digitally driven society while simultaneously striving to reduce the environmental footprint of telecommunications infrastructure [1]–[3].

Realizing network slicing, however, presents a major scientific and technological challenge. Each network slice is defined by a distinct set of requirements, formalized within its Service Level Agreement (SLA) [4]–[6]. An effective network slicing system must balance two competing objectives: ensuring isolated, customizable environments that comply with slice-specific SLAs – critical for slice tenants – while simultaneously maximizing resource utilization, a priority for the Infrastructure Provider (InP). Strict isolation prevents interference between slices but may lead to inefficient resource utilization, particularly during periods of low demand [5]. Conversely, resource sharing enhances efficiency but introduces the risk of SLA violations due to resource contention, an issue that is particularly pronounced in the Radio Access Network (RAN), where spectrum resources are inherently constrained [7]. As a result, resource allocation for RAN slicing has attracted significant attention from the research community [5], [8], [9].

Inspired by the strategy of overbooking and recognizing that network slices do not always fully utilize their allocated resources, prior works [5], [10], have proposed solutions that deliberately under-provision slices with forecasted low demand to accommodate additional slices. While such approaches can enhance profitability for Infrastructure Providers (InPs), SLA violations can occur due to the inherent uncertainty in demand forecasting. The authors of [11] and [12] also frame the problem as one of benefit maximization but employ auction-based mechanisms in which the InP functions as the seller and tenants act as bidders. While these methods prevent SLA violations when resources are plentiful, they prioritize tenants with the highest bids during periods of resource scarcity, potentially limiting access for lower-bidding tenants. Shifting towards a more tenant-centric perspective, [9] focuses on optimizing the utility of individual slices although at the expense of some overall resource utilization.

Recognizing the absence of a comprehensive solution that simultaneously ensures both slice SLA compliance and overall resource utilization efficiency, [8] introduces a hierarchical, channel-aware scheduling framework designed to enhance spectral efficiency while allocating radio resources to network slices in accordance with their respective SLAs. However, the framework guarantees SLA compliance only in a cumulative, long-term sense. In the short term – within a given scheduling window – SLA violations may still occur. Ensuring short-term compliance is critical, as it directly affects the instantaneous performance of a slice, particularly important for applications with sensitive quality-of-service requirements. Moreover, the proposed framework lacks a theoretical foundation for formally analyzing the problem and assessing the worst-case performance of its solutions. Additionally, while existing

ISBN 978-3-903176-72-0

research consistently considers SLA compliance, there is no framework for quantifying it in a way that enables verification and meaningful comparisons across different approaches.

In this paper, we formally define the problem of SLA-compliant resource allocation and develop a theoretical framework that informs the design of BLOCBIN, a resource[1] allocation method depicted in Fig. 1. BLOCBIN comprises two key components: a Compliance Manager and an RBG-Slice Allocator working in tandem to ensure both adherence to SLA constraints and efficient resource utilization. A fundamental challenge in achieving SLA compliance arises when an SLA-compliant allocation is mathematically impossible (more in § III) in a particular scheduling window. To address it, the Compliance Manager systematically monitors all allocations, allows controlled SLA violations in the particular scheduling window, and compensates the short-changed slices in the very next scheduling window. To satisfy the InP demand for resource efficiency, the RBG-Slice Allocator dynamically maps individual resource blocks to slices based on user equipment (UE) channel quality. Each slice's channel-aware intra-slice scheduler assists the RBG-Slice Allocator by prioritizing users with the best channel quality. Finally, we introduce two novel metrics designed to quantify SLA compliance over both short-term and long-term timescales, providing a rigorous basis for evaluating the effectiveness of BLOCBIN.

To summarize, we make the following key contributions:

1) We formalize the problem of SLA-compliant resource allocation and show that it is NP-Hard (§ III).
2) We introduce BLOCBIN, a heuristic for SLA-compliant and spectrum-efficient resource allocation for RAN slicing (§ IV-A).
3) We prove that BLOCBIN's efficiency in the short-term ultimately leads to fast convergence in the long-term. Moreover, we show that the period required to converge is finite (§ IV-B).
4) We introduce two novel metrics for quantifying SLA compliance and use them to evaluate BLOCBIN using trace-driven simulations. The results show that BLOCBIN performs $1.5\times$ better than state-of-the-art solutions in terms of short-term SLA compliance, displays rapid convergence in the long-term without compromising spectrum efficiency, thus achieving its goal (§ VI).

II. BACKGROUND

The architecture of 4G and 5G mobile networks consists of two main components: the Radio Access Network (RAN) and the Core Network (CN). In this section, we provide a brief overview of the RAN part, the emphasis of our work.

A. Radio Access Network (RAN)

The RAN includes User Equipment (UE) and base stations, known as evolved Node B (eNB) in 4G and next-generation Node B (gNB) in 5G. The RAN facilitates wireless communication by managing the air interface (radio resource), which includes radio resource allocation [13].

[1]RBG - Resource Block Groups, see § II.

1) Radio Resource: The air interface employs downlink and uplink user data channels as well as control channels [14]. In this work, we focus primarily on the downlink user data channel. In 4G and 5G technologies, Orthogonal Frequency Division Multiple Access (OFDMA) partitions the downlink user data channel, specifically the Physical Downlink Shared Channel (PDSCH), into multiple orthogonal subcarriers [14]–[16]. OFDMA allows UEs to access orthogonal radio resources in time and frequency, creating a 2-D grid [14], [17], as shown in Fig. 1. In the frequency domain, the fundamental unit is the subcarrier, while in the time domain, multiple units exist, including slot, subframe, and frame. There are two main resource unit structures: Physical Resource Block (PRB), the smallest allocatable unit, and Resource Block Group (RBG), consisting of multiple contiguous PRBs for efficient allocations. The size of an RBG depends on the system bandwidth; for instance, in a $20MHz$ LTE system, each RBG contains four PRBs [13], [16], [18]. 5G introduces numerology (μ), where μ ranges from 0 to 6 (as per Release 17 [19]), allowing for flexible subcarrier spacing from $15kHz$ to $960kHz$. The PRB size varies based on the selected numerology [20], [21].

Both LTE and 5G share the concept of a Transmission Time Interval (TTI)—equivalent to a subframe in LTE and a slot in 5G—which represents duration required to allocate resource units to UEs [13], [14].

For simplicity, in this paper, we define a resource unit as an RBG and the TTI length as $1\,\mathrm{ms}$.

2) Radio Resource Allocation: Resource allocation is performed at the Medium Access Control (MAC) layer of the base station and it involves distributing RBGs among UEs. A base station employs two schedulers: one for downlink and another for uplink [9], [16]. On the downlink, the scheduler applies a specific scheduling policy to determine which data to transmit, to which UE, and through which resource unit, a process we define as UE-RBG mapping. This decision is made at the beginning of every TTI [22]. To facilitate scheduling, each UE periodically estimates the channel quality for each PRB and generates a Channel Quality Indicator (CQI) report, which is sent to the base station every TTI. This report includes key metrics such as the Signal-to-Interference-plus-Noise Ratio (SINR) [14]–[16].

B. Network Slicing

Network slicing is a fundamental paradigm in 5G architecture, designed to address the 5G challenge of supporting diverse service requirements [6], [17], [23]. It enables the coexistence of multiple virtual networks (slices) with heterogeneous service demands on a shared physical infrastructure [4]–[6]. Each slice operates under a defined SLA, which specifies QoS parameters and other operational constraints. A slice is provisioned by an Infrastructure Provider (InP) and leased to a tenant, who utilizes it to serve its UEs [24], [25]. The InP must ensure SLA compliance while maintaining slice isolation, customization, and efficient resource utilization. In particular, RAN slicing involves partitioning the access network into

multiple slices while ensuring isolation, customization, and efficient utilization of the RBGs.

III. Problem Definition

Ideally, RAN slices should receive their SLA-compliant shares of RBGs (defined by their weights) at every TTI. We refer to this as *short-term compliance*. However, as discussed in Section II, the radio resource structure has strict granularity constraints. Except for specific setups, short-term compliance is not achievable in practice. To see that, consider a BS with 12 RBGs and two backlogged slices: Slice 1 and Slice 2, with respective weights of 0.7 and 0.3. This means that the target shares for Slice 1 and Slice 2 are 8.4 and 3.6 RBGs, respectively. In practice, fractional shares cannot be allocated since RBGs are the smallest unit of resources. Moreover, rounding techniques often result in undesirable or even infeasible allocations. For example:

- Rounding down, i.e., $\lfloor 8.4 \rfloor = 8$ and $\lfloor 3.6 \rfloor = 3$, underutilizes resources.
- Rounding up, i.e., $\lceil 8.4 \rceil = 9$ and $\lceil 3.6 \rceil = 4$, exceeds the available RBGs.
- Standard rounding, i.e., $\lfloor 8.4 \rfloor = 8$ and $\lceil 3.6 \rceil = 4$, results in a non-SLA-compliant allocation.

Therefore, we propose allocating RBGs in a way that ensures the target share is met on a *time-average* basis, a concept we refer to as *long-term compliance*. In the following, we formally present the problem as an optimization model.

A. Problem Formulation

Consider a gNB with G RBGs available per TTI, serving S slices. We observe the gNB operation over a time horizon consisting of T TTIs. Firstly, we define the set of binary decision variables representing the *RBG-Slice Allocation* as

$$\mathbf{x}(t) \in \{0,1\}^{S \times G}, \quad t = 1, \ldots, T, \tag{1}$$

where each variable indicates whether RBG g is allocated to slice s at TTI t (i.e., $x_{s,g}(t) = 1$) or not (i.e., $x_{s,g}(t) = 0$). Now, we formally represent the underlying requirements as optimization constraints.

Singularity Constraints: Each RBG g is allocated to exactly one slice at every TTI, i.e.,

$$\sum_{s \in [S]} x_{s,g}(t) = 1, \quad \forall g \in [G],\, t = 1, \ldots, T, \tag{2}$$

where $[n] = \{1, \ldots, n\}$, for some $n \in \mathbb{Z}_+$. A consequence of (1) and (2) is that exactly G RBGs are allocated each TTI.

Compliance Constraints: As per the SLA, in the long run, each slice s must receive its previously agreed fraction $w_s \in [0,1]$ of the gNB's RBGs, where $\sum_{s \in [S]} w_s = 1$. The average amount of RBGs allocated to each slice s over T must meet w_s. This is ensured, for each $s \in [S]$, by the following constraints

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{g \in [G]} x_{s,g}(t) = Q_s, \tag{3}$$

where we refer to $Q_s = w_s \cdot G$ as the *quantum* of slice s. The quantum Q_s of slice s represents the average of RBGs that must be allocated to slice s at every TTI.

Objective Function: Each slice's intra-slice scheduler has a specific target, e.g., throughput maximization. Given an RBG-Slice allocation $\mathbf{x}(t)$ at TTI t, we denote by $p_{s,t}(\mathbf{x}(t))$ a *potential performance* function, quantifying the performance experienced by slice s's intra-slice scheduler's policy.

Finally, we define our problem as follows

Problem 1. Our goal is to maximize the potential performance while satisfying singularity and compliance constraints, i.e.,

$$\begin{aligned} \underset{\{\mathbf{x}(t)\}_{t=1}^{T}}{\text{maximize}} \quad & \sum_{t=1}^{T} P_t(\mathbf{x}(t)) = \sum_{t=1}^{T} \sum_{s \in [S]} \alpha_s \cdot p_{s,t}(\mathbf{x}(t)) \\ \text{subject to} \quad & (1), (2), \text{ and } (3) \end{aligned}$$

where $\alpha_s, \forall s \in [S]$, normalize the different performance functions within a unified context among slices.

B. Nested Decomposition

Problem 1's components have different types of time dependency. While constraints (2) do not depend of the decisions taken in previous TTIs, constraints (3) bind the RBG-Slice allocations throughout all T TTIs. In this section, we first introduce a different formulation for Problem 1 and then we propose a decomposed view, consisting of (i) *inner* and (ii) *outer* components.

Firstly, we introduce the set of variables

$$\mathbf{b}(t) \in \mathbb{Z}_{\geq 0}^{S}, \quad 1 \leq t \leq T, \tag{4}$$

where $b_s(t)$ represents the amount of RBGs allocated to slice s at TTI t. We henceforth refer to variables $\mathbf{b}(t)$ as *bins*. Notice that, by the definition of RBG-slice allocation (1), bins can be expressed in terms of variables $\mathbf{x}$ as follows

$$b_s(t) = \sum_{g \in [G]} x_{s,g}(t), \quad \forall s \in [S], 1 \leq t \leq T. \tag{5}$$

Problem 2. Our goal is to maximize the average performance over time while satisfying singularity and compliance constraints using bins, i.e.,

$$\begin{aligned} \underset{\{\mathbf{x}(t), \mathbf{b}(t)\}_{t=1}^{T}}{\text{maximize}} \quad & \sum_{t=1}^{T} P_t(\mathbf{x}(t)) = \sum_{t=1}^{T} \sum_{s \in [S]} \alpha_s \cdot p_{s,t}(\mathbf{x}(t)) \\ \text{subject to} \quad & (1), (2), (4), \text{ and } (5), \\ & \frac{1}{T} \sum_{t=1}^{T} b_s(t) = Q_s, \quad \forall s \in [S] \qquad (6) \\ & \sum_{s \in [S]} b_s(t) = G, \quad 1 \leq t \leq T \qquad (7) \end{aligned}$$

where constraints (6) replace the original compliance constraints by using the bin definition and, for coherence, we include constraints (7) to ensure that each one of the G RBGs in each bin is assigned to a slice at every TTI. We remark that Problem 1 and 2 are equivalent.

We decompose Problem 2 into two subproblems: (i) an *inner problem*, addressing allocation decisions within each TTI, and (ii) an *outer problem*, ensuring temporal coherence of time-dependent constraints.

Let $\hat{b}_s$ be an integer representing a fixed assignment for variable $b_s(t)$ for slice s at a given TTI t. We define the inner component of our decomposition as follows

Problem 3. In the inner component, we wish to find an RBG-Slice allocation that maximizes performance at TTI t when using pre-defined bins $\hat{\mathbf{b}}$ as an input, i.e.,

$$\begin{aligned} \underset{\mathbf{x}(t)}{\text{maximize}} \quad & P_t(\mathbf{x}(t)) = \sum_{s\in[S]} \alpha_s \cdot p_{s,t}(\mathbf{x}(t)) \\ \text{subject to} \quad & \text{(1) and (2)} \\ & \sum_{g\in[G]} x_{s,g}(t) = \hat{b}_s, \quad \forall s \in [S] \end{aligned} \tag{8}$$

where constraints (8) replace (5) for a fixed bin assignment $\hat{\mathbf{b}}$.

In Problem 3, we retain the generic definition of the objective function and gather all constraints containing RBG-slice allocation variables $\mathbf{x}(t)$. Now, let $\mathbf{x}^*_{\hat{\mathbf{b}}}$ be an optimal RBG-Slice allocation for Problem 3 for a given set of bins $\hat{\mathbf{b}}$. If we denote the optimal value of the objective function for a given assignment $\hat{\mathbf{b}}$ by

$$P'_t(\hat{\mathbf{b}}) = P_t\left(\mathbf{x}(t) = \mathbf{x}^*_{\hat{\mathbf{b}}}\right). \tag{9}$$

Finally, we define the outer component of our decomposition as follows

Problem 4. In the outer component, we wish to find the values of the bins that maximize performance at every TTI and guarantees compliance to slices' RBG requirements, i.e.,

$$\begin{aligned} \underset{\{\mathbf{b}(t)\}_{t=1}^{T}}{\text{maximize}} \quad & \sum_{t=1}^{T} P'_t(\mathbf{b}(t)) = P_t\left(\mathbf{x}(t) = \mathbf{x}^*_{\mathbf{b}(t)}\right) \\ \text{subject to} \quad & \text{(4), (6), and (7)} \end{aligned} \tag{10}$$

where (10) generalizes the original objective (9) as a function of the bin variables.

Problem 4 predefines a budget of RBGs (the bins) for each slice at every TTI, while Problem 3 determines the RBG-Slice allocation. To obtain an optimal solution for Problem 2, it is necessary to solve Problem 4 by evaluating every potential solution of Problem 3. Although this decomposition has high complexity, it enables us to tackle the sub-problems separately, at different time scales. In Section IV, we exploit this characteristic to develop an algorithm for solving Problem 2.

IV. BLOCBIN

In this section, we introduce BLOCBIN, a simple and efficient algorithm designed to address Problem 1 (or equivalently, Problem 2) in practice.

A. BLOCBIN *Algorithm*

BLOCBIN is a two-level, inter-slice scheduling algorithm that leverages the decomposition introduced in Section III-B. As illustrated in Fig. 1, its main components are (i) *RBG-Slice Allocator* and (ii) *Compliance Manager*. The idea is that each one of BLOCBIN's main components will be responsible to find a solution to its respective sub-problem.

1) RBG-Slice Allocator: The RBG-slice allocator is a subroutine used to find a feasible solution for Problem 3 at every TTI. For RBG-Slice allocation, we use the greedy algorithm introduced in [8], which we refer to as GREEDY. Its simplicity and low time complexity make it a suitable candidate to handle the allocation. After computing the bins $\hat{\mathbf{b}}$ at each TTI, we input them to GREEDY and obtain the RBG-Slice allocation $\mathbf{x}^{\text{GREEDY}}$. GREEDY works as follows. First the bins for the slices at a given TTI are determined in BLOCBIN's main loop (discussed later). Then, the intra-slice scheduler of each slice s provides an estimate of the performance $p_{s,t}\left(\hat{\mathbf{x}} : \hat{x}_{s,g} = 1\right)$ if RBG g is given to it. The list of estimates is sorted and we directly assign the RBGs to the slices with the best performance until reaching their bin capacity.

2) Compliance Manager: The compliance manager is BLOCBIN's main loop and it iteratively builds a solution for Problem 4, guaranteeing compliance with slices' RBG requirements throughout the TTIs. In this part, we process each slice in rounds, where slice s maintains a "credit" mechanism controlled by variables *budget* $\beta_s(r)$ and *residual credit* $\delta_s(r)$, both evolving over rounds r. The budget $\beta_s(r)$ regulates credit allocation, later used to "acquire" RBGs, while $\delta_s(r)$ tracks any unused credit. If credits reflect slice weights, this mechanism ensures RBG allocation aligns with their resource shares by compensating for imbalances from previous rounds.

We provide BLOCBIN's pseudo-code in Algorithm 1.

Algorithm 1: BLOCBIN

Input: G, S, Time horizon T, $w_s, \forall s \in [S]$.
Output: Bins $\mathbf{b}(t) = \{b_s(t)\}_{s\in[S]}$, $\mathbf{x}(t), 1 \le t \le T$.

$\mathbf{b}(t) \leftarrow \mathbf{0},\ 1 \le t \le T$
$\delta_s(0) \leftarrow 0,\ \beta_s(0) \leftarrow 0,\ \forall s \in [S]$
$Q_s \leftarrow w_s \cdot G,\ \forall s \in [S],\ t \leftarrow 1,\ r \leftarrow 0$
while $t \le T$ **do**
  $r \leftarrow r + 1$
  for $s \in [S]$ **do**
    $\beta_s(r) \leftarrow \delta_s(r-1) + Q_s$
    while $\beta_s(r) \ge 1$ **do**
      $\beta_s(r) \leftarrow \beta_s(r) - 1$
      $b_s(t) \leftarrow b_s(t) + 1$
      if $\sum_{s\in[S]} b_s(t) = G$ **then**
        $\mathbf{x}(t) \leftarrow \text{GREEDY}(\mathbf{b}(t))$
        $t \leftarrow t + 1$
    $\delta_s(r) \leftarrow \beta_s(r)$
return $\mathbf{b}(t), \mathbf{x}(t), 1 \le t \le T$

Algorithm 1 initializes variables bins $\mathbf{b}(t)$ as well as the initial residual credit $\delta_s(0)$ and budget $\beta_s(0)$ for each slice. It also computes quantum Q_s based on slice s's weight w_s. In each round r, it initializes the budget $\beta_s(r)$ of each slice s by summing its residual credit from the previous round $\delta_s(r-1)$ and its quantum Q_s. For each slice s, RBGs are allocated until the slice's budget is exhausted or all RBGs for the TTI are assigned. Once the allocation fills G RBGs for a TTI, the GREEDY algorithm is called and finalizes the decision. The process repeats until all T TTIs are processed.

In Algorithm 1, r denotes the round index, which indicates the iterations at which each slice receives their portion of resources. Conversely, the iterator t represents the TTI and tracks the sequential RBG-Slice allocation limited to the slices' bins. While t progresses only when all G RBGs for a specific TTI are allocated, r continuously increments, ensuring that "unallocated" resources (via residual credit) carry over across rounds. This separation allows the algorithm to handle both per-TTI resource distribution and inter-TTI compliance.

B. BLOCBIN*'s Properties*

In general, resource schedulers often rely on the concept of *scheduling periods*. In the context of RBG-Slice allocation, a scheduling period is the fixed, recurring time interval within which RBGs are allocated to slices, ensuring that Problem 4's constraints are satisfied. Specifically, if T^{SP} is the duration of the scheduling period in TTIs, the following equations hold

$$\begin{aligned} &\frac{1}{T^{\text{SP}}}\sum_{t=1}^{T^{\text{SP}}} b_s^{\text{SP}}(t) = Q_s, \quad \forall s \in [S], \\ &\sum_{s\in[S]} b_s^{\text{SP}}(t) = G, \qquad 1 \le t \le T^{\text{SP}}, \end{aligned} \tag{11}$$

where $b_s^{\text{SP}}(t)$ is the assignment of RBGs to slice s at TTI t, given that the scheduling period is T^{SP}. We can define BLOCBIN's scheduling periods in terms of rounds, R^{SP}. Although the occurrence of a scheduling period may not be guaranteed in an observable time, we can still extract important properties about BLOCBIN's theoretical performance.

By the definition of scheduling period, we know that, if we run BLOCBIN for T^{SP} TTIs, our compliance constraints are satisfied and, thus, BLOCBIN outputs a feasible solution for Problem 4. However, for a general time horizon T, the bins assignments are not guaranteed to comply with the slices' requirements. We discuss next how BLOCBIN's solution converges to a feasible solution in the long run, when $T \to \infty$.

Proposition 1. Any assignment $\mathbf{x}^{\text{BB}}, \mathbf{b}^{\text{BB}}$ resulting from Algorithm 1 is a feasible solution for Problem 2 if $T \to \infty$.

Proof. We show that any solution provided by Algorithm 1 satisfies all constraints of Problem 2. For that, it suffices to prove separately that (i) the RBG-Slice Allocator, GREEDY, produces a feasible solution for Problem 3 at every TTI, and (ii) the Compliance Manager, produces a feasible solution for Problem 4, for the entire operation time T.

Firstly, we claim that GREEDY will always provide an RBG-Slice allocation $\mathbf{x}^{\text{BB}}$ that (i) guarantees slice isolation and (ii) is bounded by the bins $\mathbf{b}^{\text{BB}}$ [8]. Even though the resulting RBG-Slice assignment may be sub-optimal for Problem 2, it always satisfies Problem 3's constraints. Now, if $\mathbf{b}^{\text{BB}}$ is a feasible assignment, so is $\mathbf{x}^{\text{BB}}$. In what follows, we focus on showing that BLOCBIN is capable of finding a feasible solution for Problem 2 in terms of bins $\mathbf{b}^{\text{BB}}$, by also satisfying Problem 4's constraints:

- As per its definition (4), the bin of each slice at every TTI is a non-negative integer.
- In lines 14-16 of Algorithm 1, we ensure that we only move to the next TTI if the number of assigned bins meets the total number of RBGs G, which satisfies (7).
- The last step is to show BLOCBIN satisfies the compliance constraints (6).

Consider the number of RBGs assigned to slice s at every round r. We denote it by $b_s(r)$ and it can be seen as a version of the bins adapted to BLOCBIN's rounds. At every round r of BLOCBIN, we compute slice s's total budget as

$$\beta_s(r) = \delta_s(r-1) + Q_s. \tag{12}$$

This is defined in line 10 as the residual budget of the previous round plus the slice's quantum. Then, we iteratively assign RBGs in the loop defined in lines 11-16. At the end of the round, the total number of assigned bins to slice s is

$$b_s(r) = \lfloor \beta_s(r) \rfloor = \beta_s(r) - \delta_s(r), \tag{13}$$

where $\delta_s(r)$ is the new residual budget of slice s for current round r produced in this process. In the end, we can combine (12) and (13) to obtain the following recursive definition for the bin of slice s at round r

$$b_s(r) = \delta_s(r-1) + Q_s - \delta_s(r). \tag{14}$$

Now, if we consider the total amount of RBGs assigned to slice s at every round by summing (14) throughout the entire operation time horizon, we obtain

$$\sum_{r=1}^{R} b_s(r) = \sum_{r=1}^{R} \delta_s(r-1) + Q_s - \delta_s(r), \tag{15}$$

where R is the number of rounds needed to assign RBGs for T TTIs. This is a telescoping series, where all intermediate $\delta_s(r)$ terms are canceled out and only the first term, $\delta_s(0)$, and the last term, $\delta_s(R)$, remain in the resulting equation.

We develop the sum (15) as follows

$$\sum_{r=1}^{R} b_s(r) = R \cdot Q_s + \delta_s(0) - \delta_s(R) \Leftrightarrow \tag{16}$$

$$\Leftrightarrow \frac{1}{R}\sum_{r=1}^{R} b_s(r) = Q_s - \frac{\delta_s(R)}{R}. \tag{17}$$

We recall that the first term $\delta_s(0) = 0, \forall s \in [S]$ as there are no residual budgets before the first round. In general, we cannot state anything about the precise value of $\delta_s(R)$, in (17), as it fluctuates within $[0, 1)$ depending on the time

horizon in terms of rounds R. However, notice that if $R \to \infty$, then $\frac{\delta_s(R)}{R} \to 0$, and, from (17), we obtain

$$R \to \infty \Rightarrow \frac{1}{R} \sum_{r=1}^{R} b_s(r) \to Q_s. \tag{18}$$

In other words, it means that, the longer we run BLOCBIN, the smaller is the impact of the final round's residual budgets. At the same time, the potentially infinite occurrences of scheduling periods will drift the average bin towards the quantum.

Now, we move back to the time domain in terms of TTIs. If we have T TTIs, we need at least as many rounds R to determine the bins for each TTI. This way, the number of TTIs is always bounded by the number of rounds. According to our initial hypothesis, we have that $T \to \infty$, then, naturally, the number of BLOCBIN's rounds to generate the T bin assignments also tends to infinity, i.e., $R \to \infty$. Then, the analysis developed for BLOCBIN's rounds also holds for TTIs when $T \to \infty$, leading to the following result

$$T \to \infty \Rightarrow \frac{1}{T} \sum_{t=1}^{T} b_s^{\mathrm{BB}}(t) \to Q_s. \tag{19}$$

Therefore, BLOCBIN is capable of finding bins assignments $\mathbf{b}^{\mathrm{BB}}$ that converge to the slices' quanta when $T \to \infty$, which satisfies the compliance constraints (6).

Finally, because constraints (4), (6), and (7) are satisfied when $T \to \infty$, we conclude that BLOCBIN provides feasible solutions for Problem 2. □

Although the feasibility of a solution provided by BLOCBIN is only demonstrated asymptotically, when $T \to \infty$, we remark that, in practice, scheduling periods occur in early stages of the algorithm's execution. We discuss about convergence in Section VI.

V. SIMULATION SETUP

A. Hardware and Software Environment

(i) Simulation Platform: We conducted our simulations on Ubuntu 22.04 LTS operating system with Intel Core i7 5.2 GHz processor and 64 GB of RAM.

(ii) LTE-Sim Simulator: LTE-Sim is an open-access, system-level, discrete-event network simulator written in C/C++. It is widely used to evaluate the performance of LTE networks [26], [27], particularly for testing new Radio Resource Management schedulers [28]–[30]. In our experiments, we utilize an enhanced version [8] that incorporates 5G functionalities, such as network slicing abstractions and hierarchical scheduling.

(iii) Channel Traces: In our simulations, we use real channel traces, as in [8]. The original traces, obtained from LTScope [31], are collected by measuring LTE signal strength (SNR over time and across sub-carriers) for major U.S. mobile carriers. [8] extended these measurements in the 5G domain by concatenating five randomly-selected, distinct measurements of the 20 MHz LTE bandwidth to generate measurements over a 100 MHz 5G bandwidth.

B. Network Architecture

The network architecture features one cell with a single gNB connected to a sole gateway and multiple UEs. The cell has a bandwidth of 100 MHz (512 PRBs) and a radius of 1 km. The UEs are randomly placed around the gNB and remain static during the simulations. The gateway is the source of all downlink traffic to the UEs.

Each UE is linked to a specific network slice and has a unique channel quality trace file with CQI values for each of the 512 PRBs at each TTI. UEs send their CSI reports to the gNB every 1 ms. Each UE has a source IP address and receives data using the User Datagram Protocol (UDP), the only transport protocol supported by the LTE-Sim simulator.

C. Traffic Model

In the simulation setup, we considered three types of traffic: backlogged flow (infinite buffer), internet flow (heavy-tail distribution), and trace-based video flow. The average bitrate to the gNB is defined in the slice configuration. For a slice with a 9 Mbps aggregate data rate, the per-user flow rate is calculated by dividing this rate by the number of UEs. In scenarios involving multiple flows, priority is assigned based on the order of flow creation.

For the trace-based video flow, we consider a video sequence from the Foreman clip. This video sequence has been encoded using H.264 compression at an average bitrate of 1.28 Mbps. Each entry in the trace file includes the packet size of the data frame, the type of frame, the index number of the frame, and the timestamp.

VI. SIMULATION RESULTS

In this section, we evaluate BLOCBIN and assess its ability to fulfill its objective of making SLA-compliant allocations while maintaining high spectrum efficiency. We compare BLOCBIN to two state-of-the-art scheduling schemes, Network Virtualization Substrate (NVS) [9] and RadioSaber [8], using their existing LTE-Sim implementations.

A. Performance Metrics

To evaluate the spectrum efficiency we use throughput, whereas to quantify SLA compliance we introduce two normalized metrics: *short-term* and *long-term* compliance. These metrics are designed to quantify with a simple-to-understand value between 0 and 1 the difference between the resources a slice is entitled to receive according to the SLA and those it is actually allocated – thus serving as a proxy for SLA compliance.

We define the *compliance index* for slice s at TTI t as

$$c_s(t) = 1 - \left(Q_s - b_s(t)\right)/Q_s, \tag{20}$$

where Q_s is the quantum of a target slice s and $b_s(t)$, the bin, is the actual share of the slice s at a given TTI t. The compliance index quantifies the difference between what a slice should receive, i.e., the quantum, and what it is actually allocated, i.e., the bin, on a given TTI. A higher value of $c_s(t)$ yields a higher SLA compliance for slice s at TTI t.

(i) Short-Term Compliance: We evaluate SLA compliance over all the slices in a particular TTI by applying the slice compliance to the fairness index as follows

$$C(t) = \frac{(\sum_{s\in[S]} c_s(t))^2}{S\sum_{s\in[S]} c_s^2(t)}, \tag{21}$$

where $c_s(t)$ is defined in (20)

(ii) Long-Term Compliance: This metric serves as the long-term analog to (21) and measures the cumulative performance of algorithms until TTI t, i.e.,

$$\bar{C}(t) = \frac{\left(\sum_{s\in[S]} \bar{c}_s(t)\right)^2}{S\sum_{s\in[S]} \bar{c}_s^2(t)}, \tag{22}$$

where $\bar{c}_s(t) = \sum_{t'=1}^{t} c_s(t')$.

B. Scenario 1: Always-Active Slices

We evaluate the performance of BLOCBIN, RadioSaber, and NVS as the slices continuously demand resources at each TTI. We configure the system with 50 slices, each using a PF scheduler, similarly to RadioSaber [8]. In each slice, four UEs receive backlogged flows from the gateway. The simulation time is set to 10s, corresponding to 10^4 TTIs.

1.1) Uniform Weight Distribution: First, we consider the assigned weight for each of the 50 slices to be $w_s = 0.02$. Fig. 2a presents the short-term compliance, $C(t)$, measured at every TTI. The data shows that BLOCBIN, with average compliance of 80%, significantly outperforms RadioSaber's 53% and NVS's 2%. This improvement is further underlined by the algorithms' dispersion over time, where BLOCBIN benefits from its compensation methods. It achieves a smaller dispersion, with a standard deviation of 9%, compared to RadioSaber's 15%. In contrast, NVS's lack of a compensation strategy results in no standard deviation, as it is confined to serving a single slice per TTI, severely limiting its short-term SLA compliance. Furthermore, the data shows that BLOCBIN exhibits a periodic pattern of compliance. This behavior is linked to its scheduling period property, as elaborated in Section IV-B, which spans 26 TTIs in this case. After this period, the value of $\mathbf{b}^{\text{SP}}$ is reiterated throughout the simulation. Moreover, BLOCBIN's performance is strengthened by aligning the value of $b_s(t)$ with Q_s for each slice at every TTI, effectively addressing unfairness in subsequent time intervals. This approach significantly improves its compliance compared to NVS and RadioSaber without compromising its throughput performance, as shown in Fig. 2c.

Fig. 2b depicts the long-term compliance performance of the three schemes. BLOCBIN demonstrated the most rapid convergence, followed by RadioSaber, with NVS exhibiting the slowest convergence. By the end of the simulation, all slices across the three schemes achieved full compliance (100%). This result supports the observations in Section IV-B, highlighting BLOCBIN's fast convergence capabilities when scheduling periods occur in the initial stages of the algorithm's execution.

TABLE I: Scenario 2's groups of slices description

Group	Slices	Weight	Scheduler	Traffic Pattern
1	20	0.0275	PF	Constant
2	10	0.01	PF	12-Mbps Heavy-Tail
3	10	0.02	PF	2- and 9-Mbps Heavy-Tail
4	10	0.015	M-LWDF	1280-kbps video flow

1.2) Variable Weight Distribution: This case includes two distinct groups of slices: the first group is assigned $w_s = 0.01$, while the second group is assigned $w_s = 0.03$. Fig. 3a shows a slight performance decline for BLOCBIN and RadioSaber in terms of short-term compliance compared to the previous scenario. This reduction can be attributed to the differing slice weights, making the distribution of resources among the two groups in a single TTI more challenging. Nevertheless, with 72% average compliance rate, BLOCBIN significantly outperforms RadioSaber's 45%. Furthermore, Fig. 3b shows that BLOCBIN converges the fastest, achieving full long-term compliance within 6 TTIs. In contrast, RadioSaber reaches the 100% target around 100 TTIs, followed by NVS. As in the previous scenario, Fig. 3c shows that this improvement in SLA compliance does not compromise spectrum efficiency.

C. Scenario 2: Diverse Traffic

In this scenario, we analyze the performance of the three scheduling schemes under varying traffic conditions. Our simulations involve 50 slices partitioned in groups as shown in Table I.

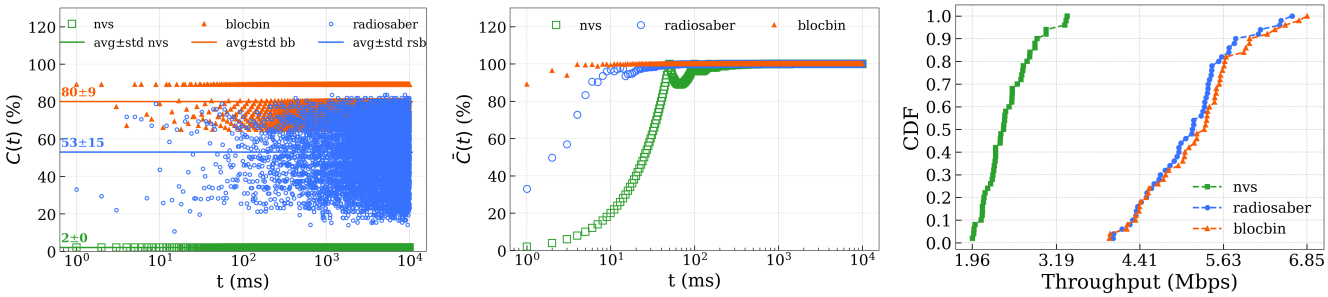
Fig. 4 shows that, even in this challenging scenario, BLOCBIN significantly outperforms RadioSaber and NVS in terms of SLA compliance without compromising throughput performance. In particular, the average short-term compliance for BLOCBIN is $79\pm5\%$, while RadioSaber's only $45\pm17\%$. A careful analysis of Fig. 4a shows that BLOCBIN deviates from the standard periodic performance displayed in the previous scenarios. This can be attributed to the inactivity of some slices during the simulation. Consequently, BLOCBIN's compliance manager redistributes resources from these inactive slices to active ones, enhancing resource utilization. Similarly, RadioSaber's performance drop suggests an adaptive response to inactive slices.

VII. Related Work

Efficiently sharing network resources among isolated slices to avoid SLA violations is a crucial problem in network slicing. While numerous solutions have been proposed, they generally fall into one of three categories: channel-aware allocation, slice overbooking, and auction-based mechanisms.

A. Channel-Aware Approach

RadioSaber [8] presents a hierarchical channel-aware scheduling framework aimed at enhancing spectral efficiency while allocating RBGs to slices based on the SLA-defined percentages. It integrates a two-level inter-slice and intra-slice scheduling. The inter-slice scheduler, implemented by the InP, initially allocates resources to slices so as to comply

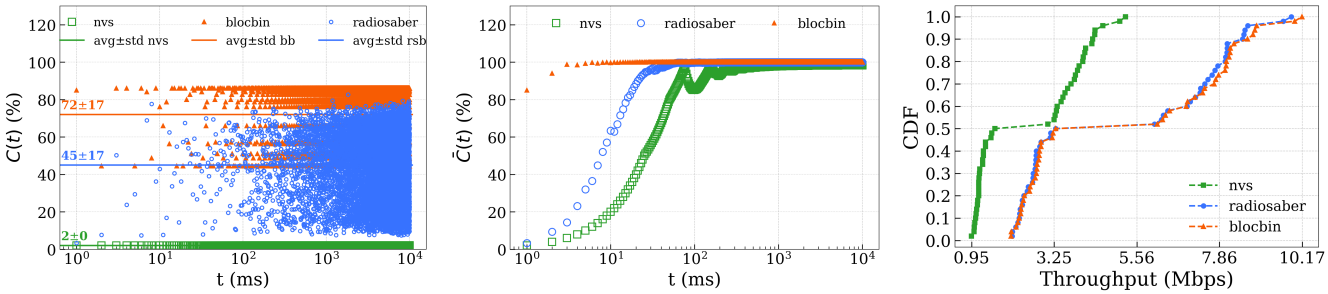


(a) Short-term Compliance $C(t)$ over time (TTI), where dispersion is mean $\pm$ standard deviation.

(b) Long-term Compliance $\bar{C}(t)$ over time (TTI). Target compliance is $\bar{C}(t) = 100\%$.

(c) Empirical CDF of UEs' throughput, showing the fraction of UEs with throughput $\leq$ a given value

Fig. 2: Experimental results for Scenario 1.1, where all 50 slices have $w_s = 0.02$.

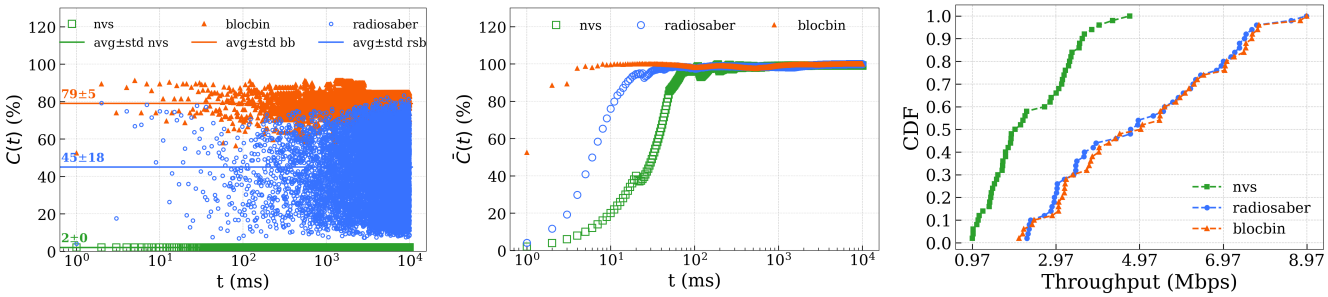


(a) Short-term Compliance $C(t)$ over time (TTI), where dispersion is mean $\pm$ standard deviation.

(b) Long-term Compliance $\bar{C}(t)$ over time (TTI). Target compliance is $\bar{C}(t) = 100\%$.

(c) Empirical CDF of UEs' throughput, showing the fraction of UEs with throughput $\leq$ a given value

Fig. 3: Experimental results for Scenario 1.2, where 25 slices have $w_s = 0.01$ and 25 slices have $w_s = 0.03$.



(a) Short-term Compliance $C(t)$ over time (TTI), where dispersion is mean $\pm$ standard deviation.

(b) Long-term Compliance $\bar{C}(t)$ over time (TTI). Target compliance is $\bar{C}(t) = 100\%$.

(c) Empirical CDF of UEs' throughput, showing the fraction of UEs with throughput $\leq$ a given value

Fig. 4: Experimental results for Scenario 2, where slices are grouped according to different traffic patterns as in Table I.

with the SLA, followed by a greedy allocation of RBGs to eligible users according to channel quality. The intra-slice scheduler, implemented by the tenant, recommends UE-RBG mappings to the inter-slice scheduler. While RadioSaber is efficient in terms of spectral efficiency it only offers long-term SLA compliance, overlooking short-term compliance. In contrast, our solution considers short-term and long-term compliance, demonstrating that achieving both is possible without sacrificing spectrum efficiency.

The work in [9] focuses on radio resource virtualization for WiMAX systems. The proposed scheme aims at optimizing the utility of individual slices while maintaining overall system performance at the inter-slice level and allows slices to customize their intra-slice schedulers. The proposed inter-slice scheduler was used in other works, such as [16]. Although the inter-slice scheduler offers SLA compliance, it is not channel-aware, limiting its spectrum efficiency. In contrast, our proposed solution provides both SLA compliance and spectrum efficiency.

B. *Overbooking Provisioning Solutions*

Works like [10] and [5] adopt a different philosophy. They propose overbooking strategies that involve intentionally violating SLA by underprovisioning slices to accommodate

additional slices. These strategies seek to maximize profits for InPs while reducing SLA violations that can arise from overly optimistic slice demand forecasts. [5], for example, offers up to a 300% profit for infrastructure providers. Nevertheless, SLA violations, even reduced, can be a concern for many tenants.

C. Auction-based Mechanisms

[11] and [12] present hierarchical auction-based solutions where the InP acts as the seller and tenants as bidders at the inter-slice level, while tenants and subscribers fulfill these roles at the intra-slice level. Although [11] allows slices to customize their user-level strategy, it does not consider slice-level SLA compliance. [12] introduces a fairness strategy where all the bids are accepted when resources are abundant, but only accepts tenants with the highest bids during scarcity. However, this approach may result in starvation during resource shortages. In contrast, BLOCBIN proposes a well-defined fairness model that provides fixed guarantees to tenants.

VIII. CONCLUSION

We introduce BLOCBIN, a channel-aware heuristic for resource allocation in RAN slicing that proportionally distributes radio resources based on SLA-defined shares, aiming to enhance throughput performance. BLOCBIN is one of the few works that addresses the short-term compliance issue, which is critical for slice performance. Simulation results demonstrate that our approach significantly outperforms state-of-the-art methods regarding both short- and long-term SLA compliance, all while maintaining high throughput. BLOCBIN's simplicity and effectiveness are emphasized through its SLA conformity, ease of implementation, and support for slice isolation and customization. Additionally, it efficiently adapts to scenarios with inactive slices, thereby optimizing resource utilization and further improving throughput performance.

ACKNOWLEDGEMENTS

This work was supported in part by ANR under the France 2030 program, grant NF-NAI: ANR-22-PEFT-0003.

REFERENCES

[1] A. S. D. Alfoudi, S. H. S. Newaz, A. Otebolaku, G. M. Lee, and R. Pereira, "An Efficient Resource Management Mechanism for Network Slicing in a LTE Network," *IEEE Access*, 2019.

[2] U. Gupta, Y. G. Kim, S. Lee, J. Tse, H.-H. S. Lee, G.-Y. Wei, D. Brooks, and C.-J. Wu, "Chasing Carbon: The Elusive Environmental Footprint of Computing," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 2021.

[3] A. Bellin, F. Granelli, and D. Munaretto, "A measurement-based approach to analyze the power consumption of the softwarized 5G core," *Computer Networks*, May 2024.

[4] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions," *IEEE Commun. Surveys Tuts.*, 2018.

[5] S. Alcalá-Marín, A. Bazco-Nogueras, A. Banchs, and M. Fiore, "kaNSaaS: Combining Deep Learning and Optimization for Practical Overbooking of Network Slices," in *ACM MobiHoc*, Oct. 2023.

[6] A. Banchs, G. de Veciana, V. Sciancalepore, and X. Costa-Perez, "Resource Allocation for Network Slicing in Mobile Networks," *IEEE Access*, 2020.

[7] Q. Liu and T. Han, "VirtualEdge: Multi-Domain Resource Orchestration and Virtualization in Cellular Edge Computing," in *IEEE ICDCS*, Jul. 2019.

[8] Y. Chen, R. Yao, H. Hassanieh, and R. Mittal, "Channel-Aware 5g RAN slicing with customizable schedulers," in *USENIX NSDI*, Apr. 2023.

[9] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan, "NVS: a substrate for virtualizing wireless resources in cellular networks," *IEEE/ACM Trans. Netw.*, Oct. 2012.

[10] J. X. Salvat, L. Zanzi, A. Garcia-Saavedra, V. Sciancalepore, and X. Costa-Perez, "Overbooking network slices through yield-driven end-to-end orchestration," in *ACM CoNEXT*, Dec. 2018.

[11] Y. K. Tun, N. H. Tran, D. T. Ngo, S. R. Pandey, Z. Han, and C. S. Hong, "Wireless Network Slicing: Generalized Kelly Mechanism-Based Resource Allocation," *IEEE J. Sel. Areas Commun.*, Aug. 2019.

[12] L. M. M. Zorello, K. Eradatmand, S. Troia, A. Pattavina, Y. Zhang, and G. Maier, "Auction-based network slicing for 5G RAN," in *IEEE NetSoft*, Jun. 2023.

[13] D. Zhou, N. Baldo, and M. Miozzo, "Implementation and validation of LTE downlink schedulers for ns-3," in *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, ser. SimuTools '13, Mar. 2013.

[14] S. A. AlQahtani and M. Alhassany, "Comparing different LTE scheduling schemes," in *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, Jul. 2013.

[15] C. Sexton, N. Marchetti, and L. A. DaSilva, "Customization and Trade-offs in 5G RAN Slicing," *IEEE Commun. Mag.*, Apr. 2019.

[16] X. Foukas, M. K. Marina, and K. Kontovasilis, "Orion: RAN Slicing for a Flexible and Cost-Effective Multi-Service Mobile Network Architecture," in *ACM MobiCom*, Oct. 2017.

[17] A. Mamane, M. Fattah, M. E. Ghazi, M. E. Bekkali, Y. Balboul, and S. Mazer, "Scheduling Algorithms for 5G Networks and Beyond: Classification and Survey," *IEEE Access*, 2022.

[18] 3GPP, "LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures (Release 10)," 2014.

[19] ——, "5G; NR; Physical channels and modulation (Release 17)," 2022.

[20] G. Barb, M. Otesteanu, F. Alexa, and F. Danuti, "OFDM Multi-Numerology for Future 5G New Radio Communication Systems," in *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Sep. 2020.

[21] Y. Varun, K. Syam Chandran, and C. Ali, "Inter-Numerology Interference Reduction Based on Precoding for Multi-Numerology OFDM Systems," in *IEEE 5G World Forum*, Sep. 2020.

[22] G. Piro, L. A. Grieco, G. Boggia, F. Capozzi, and P. Camarda, "Simulating LTE Cellular Systems: An Open-Source Framework," *IEEE Trans. Veh. Technol.*, Feb. 2011.

[23] Qualcomm, "Future of 5G," Feb. 2020.

[24] P. Caballero, A. Banchs, G. De Veciana, and X. Costa-Pérez, "Network Slicing Games: Enabling Customization in Multi-Tenant Mobile Networks," *IEEE/ACM Trans. Netw.*, Apr. 2019.

[25] M. I. Kamel, L. B. Le, and A. Girard, "LTE Wireless Network Virtualization: Dynamic Slicing via Flexible Scheduling," in *IEEE VTC*, Sep. 2014.

[26] R. Subramanian, P. Ghosal, S. Barua, S. Xing, S. L. Cong, H. Al Kim, and K. Sandrasegaran, "Survey of LTE Downlink Schedulers Algorithms in Open Access Simulation Tools NS-3 and LTE-SIM," *International Journal of Wireless & Mobile Networks (IJWMN)*, Apr. 2015.

[27] R. Ahmad, E. A. Sundararajan, and A. Khalifeh, "A survey on femtocell handover management in dense heterogeneous 5G networks," *Telecommunication Systems*, Dec. 2020.

[28] G. Piro, L. A. Grieco, G. Boggia, R. Fortuna, and P. Camarda, "Two-Level Downlink Scheduling for Real-Time Multimedia Services in LTE Networks," *IEEE Trans. Multimedia*, Oct. 2011.

[29] E. Liotou, D. Tsolkas, N. Passas, and L. Merakos, "Quality of experience management in mobile cellular networks: key issues and design challenges," *IEEE Commun. Mag.*, Jul. 2015.

[30] I.-S. Comşa, S. Zhang, M. E. Aydin, P. Kuonen, Y. Lu, R. Trestian, and G. Ghinea, "Towards 5G: A Reinforcement Learning-Based Scheduling Solution for Data Traffic Management," *IEEE TNSM*, Dec. 2018.

[31] Y. Xie, F. Yi, and K. Jamieson, "PBE-CC: Congestion Control via Endpoint-Centric, Physical-Layer Bandwidth Measurements," in *ACM SIGCOMM*, Jul. 2020.