Received July 8, 2019, accepted July 15, 2019, date of publication July 18, 2019, date of current version August 6, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2929732

Highly Flexible RAN Slicing Approach to Manage Isolation, Priority, Efficiency

DANIA MARABISSI, (Senior Member, IEEE), AND ROMANO FANTACCI, (Fellow, IEEE)
Department of Information Engineering, University of Florence,50139 Firenze, Italy

Corresponding author: Dania Marabissi (dania.marabissi@unifi.it)

This work was supported by the project GAUChO—A Green Adaptive Fog Computing and Networking Architecture through the MIUR Progetti di Ricerca di Rilevante Interesse Nazionale (PRIN) Bando 2015 under Grant 2015YPXH4W-004.

ABSTRACT The evolution toward 5G is driven by the need of providing a wide range of services differing on needed network functionalities, performance requirements, type of devices, and going beyond the human-type communications. Such a wide variety of requirements cannot be always met through a common network setting, hence, high network flexibility and scalability are required. Network slicing allows the operation of multiple end-to-end logical networks on a common physical infrastructure: each network slice is tailored to best support a specific service. Network slicing on the radio access network (RAN) domain is challenging. In order to manage the scarce radio resources, RAN slicing requires flexibility, efficient resource sharing, and customization. Hence, dynamic resource management presents unique challenges, it has to take into account different issues that can also be in contrast with each other. This paper proposes a two-layer scheduler for an efficient and low complexity RAN slicing approach in actual systems. It is shown that simply setting some parameters it is possible to achieve different trade-offs between isolation and efficiency, allowing the management of priority and customization. The performance of the proposed method has been compared with other benchmark approaches to show good behavior and the flexibility of the proposed approach.

INDEX TERMS RAN slicing, resource allocation, 5G.

I. INTRODUCTION

The evolution toward 5G is not driven only by the need of scaling up and improving the efficiency of current mobile networks, but also providing a wide range of services differing on needed network functionality (e.g., security, mobility, radio resource management), performance requirements (e.g., latency, data rate, reliability), type of devices and going beyond the human-type communications. Fifth Generation Public Private Partnership (5GPPP) has identified three major use cases [1]: *enhanced mobile broadband* (eMBB) that requires high data rate and low latency in wide areas to improve current mobile services; *massive machine-type communications* (mMTC) whose main characteristic is providing connectivity to a massive number of devices; *ultra-reliable and low-latency communications* (URLLC) for time-sensitive critical services that require high reliability. This high variety of requirements cannot be always met through a common network setting, the current *one-size-fits-all* network architecture is not more efficient, but high network flexibility and scalability are required. For these reasons *network slicing* is considered one of the pillars of 5G systems [2]–[4]. It allows the operation of multiple end-to-end logical networks (*network slices*) on a common physical infrastructure where each logical network is tailored to provide a particular system behavior to best support specific services and/or provide a particular tenant with a given level of guaranteed network resources. A network slice is composed of a collection of network functions and specific radio access technology settings, and should be isolated from other slices with independent control and management. This can be achieved by means resource virtualization and logical partition, that allow full isolation of the underlying physical resources (processing, storage, network, and radio) among slices and the ability of supporting different types of control operations depending on the service requirements.

Network Slicing encompasses both Core Network (CN) and Radio Access Network (RAN): both parts need to be flexibly sliced into several overlaid instances serving different types of users, devices and use cases. In [5] an analysis

The associate editor coordinating the review of this manuscript and approving it for publication was Kathiravan Srinivasan.

is provided on trade-offs between customization, efficiency, and complexity in network slicing, by evaluating the impact of resource allocation at different network points.

RAN slicing is very challenging and at an early stage of investigation. Indeed, while at CN level it is possible to scale up the network by adding more hardware resources, RAN has to face with spectrum scarcity. This is a strong limitation especially if fixed portions of radio resources are dedicated to individual slices, thus limiting the potential multiplexing gain. Hence, it is needed an abstraction of radio resources, that must be managed by means of new algorithms and solutions.

This paper proposes a low complexity and high flexible scheme to find the suitable trade-off between isolation, efficiency, priority and customization in dynamic radio resources assignment among slices. We would like to underline that, in general the concept of *isolation* refers to the separation of all the network functions and resources. However, here we focus on RAN, and in particular, on radio resources slicing, hence, we refer to the isolation in terms of dedicated radio resources.

II. RELATED WORKS

Slicing the RAN presents unique challenges due to the inherent broadcast nature of the radio channel, its random fluctuations and the potential influence that any transmitter may have on any receiver: isolation among slices is not straightforward. More in details, one of the main aspects of RAN slicing is the division of the radio resources among slices. This can be based on several criteria such as bandwidth, number of resources, Service Level Agreements (SLAs), interference, traffic load or a combination of these, and has to transform such criteria in a number of physical resources to be allocated to each slice trying to be fair and satisfy their requirements. Moreover, a suitable trade-off between isolation and multiplexing gain must be reached. RAN slicing solutions adopted in current networks, are based on a simplified concept, where a fixed portion of radio resources is *dedicated* to a network slice, thus separating and isolating slices in terms of control and user plane traffic, scheduler and physical resources (each slice has its own RAN functions instances) [6]. Hence, sharing is limited to the hardware (digital processing modules, antennas, backhaul/fronthaul links, etc.). These solutions are easy to realize and assure isolation among slices through the static fragmentation of the spectrum, but are inefficient, as resources still remain dedicated to a slice even if they are not used. In 5G systems the concept of network slicing will evolve toward a more *flexible and dynamic sharing* thanks to the use of network resources virtualization. Radio resources should be dynamically shared among different network slices that use common control plane, physical layer and scheduler. Toward this goal it is needed to exploit statistical scheduling of physical resources that improves resources utilization but on the other side makes isolation and Quality of Service (QoS) guarantee more challenging. Dynamic RAN sharing solutions are currently gaining more attention even if are not widely investigated yet. Indeed, main works in the literature present architectural proposals, conceptual ideas or identified problems without detailed solutions [3], [7], [8].

A. SINGLE-LAYER SCHEDULER

Among the proposed solutions, a few are based on a *single-layer scheduler*, such as in [9]–[15]. In particular, in [9] different service providers bid on behalf their users for network resources in a sequential auction, and the infrastructure provider decides the final user allocation given the achievable rate region. Differently, in [10] resources are iteratively assigned to the users in order to increase their level of satisfaction that is differently defined for each slice. In [11] at each iteration a slice and a user are selected according to a proportional fair (PF) approach: the selected slice picks the physical resource having the best channel response, and assigns it to the selected user. An application-oriended RAN-sharing approach is proposed in [12], where the goal is to adapt the RAN to the applications' needs. The multi-objective problem, is solved by means a sub-optimal approach based on the barrier method. An analysis of the share-constrained proportional allocation mechanism is provided in [13]. In particular, each slice has a given amount of resources that depends on its *sharing level*, and customizes its own users' allocation reacting to the allocation of other slices, so as to maximize its own utility. In order to maximize the resource utilization with a constraint on the outage probability, [14] proposes the use of an offline reinforcement learning followed by a low-complexity heuristic.

In general, single-layer approaches require complex multi-service schedulers, and may incur a substantial overhead and complexity. Moreover, slices isolation and customization are more difficult. A single-layer scheduler solution, with affordable complexity is proposed in [15] where both resources allocation and users association problems are considered. The two problems are solved separately, and in particular, the resource assignment follows a PF approach: each user receives a fraction of resources that is proportional to the sharing factor of its slice normalized by the number of users of the slice.

B. TWO-LAYER SCHEDULER

In general, the most promising approach seems to be using a *two-layer scheduler*: one layer is used to determine the amount of resources for each slice (*inter-slice scheduler*), while the other is specific for each slice and allocates resources to end users (*intra-slice scheduler*). A proof-of-concept of a RAN slicing system that controls functions and resources of the underlying RAN is presented in [16]. In particular, the virtualization manager performs inter-slice resource partitioning and radio resource abstraction. However, details on how resources are partitioned and assigned to the users are not provided.

In [7], [17], intra-slice schedulers assign virtual resources to their users, then the inter-slice scheduler maps virtual resources into physical ones. Each slice has a predefined amount of allocated resources, but those left unused by a slice can be allocated to other slices to avoid waste of resources. Differently, in [18] first resources are assigned to slices with a round robin (RR) approach, and then the number of resources per slice is adjusted on a second step according to the actual average rates and target rates of the slices. In [19] a high-level entity allocates a portion of resources to each slice based on a specific contract, and on a prediction on the future requests of each slice. The paper does not provide details on how slices' requests are estimated, and how users' requests are taken into account. Similarly, [20] introduces a Network Virtualization Substrate which operates on the top of a MAC scheduler with the aim of allocating resources to slices allowing the coexistence of bandwidth and resource-based slicing. Also in this case, how slices' requests are determined, and how users' requests are managed is not specified. An auction game is proposed in [21] to allocate resources to slices, but intra-slice scheduler is not described. In particular, different kinds of resources are considered (radio, computation, storage) for which the price is calculated taking into account slices' requirements, then the competitive auction mechanism is used to achieve the optimal resource allocation. A stackelberg game is instead used at both scheduling layers in [22]. In [23] first resources are fairly divided among slices according to their priority and load, then each slice allocates resources to its users to maximize the utility. Differently, in [24] a bankruptcy game based algorithm is proposed to allocate resources to slices taking into account typical QoS parameters that characterize each slice when there is a lack of resources. The level of user's satisfaction is considered as utility. In [25], [26] the resource shortage for guaranteed services is avoided by means suitable slice admission control policies.

C. CONTRIBUTE

As already stated an actual RAN sharing approach has to take into account different aspects that can be in conflict with each other. In general, small flexibility would limit efficiency and customization, but too much flexibility might result in unnecessary complexity. This paper proposes a two-layer scheduler for an efficient and low complexity RAN slicing approach in actual systems. Main contributes are:

- differently from previous papers, where a single aspect (maximum isolation, maximum efficiency, minimum bandwidth, etc.) is considered, here, different aspects, suitably weighted, are simultaneously taken into consideration. Hence, the same algorithm can be used in different scenarios with different aims. In particular, we refer here to *radio resources isolation*, that is generally solved by assuming a reserved portion of spectrum, thus leading to inefficiency and waste of resources. Here isolation is differently managed, and can be suitably set.
- The high flexibility of the proposed algorithm is achieved simply varying few parameters. Moreover, slices' customization is guaranteed.
- Even if the proportional fair resource allocation is widely investigated, its application to RAN slicing and related challenges have not been studied yet. In particular, here resource management is split in two parts for solving inter and intra slice allocation. The main contribute of the paper is on the inter-slice scheduler that allows to manage different parameters as isolation, priority and traffic requests. The intra-slice scheduler is only an example to show the slices' customization, but different approaches could be used depending on the slices owner preferences.
- Differently from previous approaches, the algorithm is based on specific users' QoS requirements, and hence, on the number of physical resources blocks (PRB) requested by each user. Most of the papers consider the utility function maximization, but do not refer to the specific users' requests, and do not consider the actual discretization of radio resources.
- The proposed approach presents low complexity, and hence, it is easily scalable. Indeed the complexity increases linearly with the number of slices (inter-slice scheduler) and with the number of users (intra-slice scheduler).

Performance is given in comparison of benchmark methods to show the effectiveness of the proposed approach.

The paper is organized as follows. First the system model and the requirements of network slices are presented in Sect. III, then the RAN slicing problem and its solution are presented in Sect. IV. Benchmark approaches are described in Sect. V, while Sect. VI shows the effectiveness and the flexibility of the proposed scheme presenting the numerical results. Finally, conclusions are drawn in Sect. VII.

III. SYSTEM MODEL

We consider a Base Station (BS) that serves a set, $\mathcal{K}$, of active users that belong to S different service groups mapped on S different network slices. We refer with $\mathcal{S}$ to the set of network slices and with $\mathcal{K}_s \in \mathcal{K}$ the set of users associated to the s-th slice. These are randomly distributed in the area according to a point Poisson process, and the mean number of users in s-th slice is K_s.

Slices and users are characterized by QoS parameters, and physical radio resources must be allocated across the slices trying to meet such requirements and inter-slice isolation. In particular, in 5G systems radio resources still will be organized in PRB as in 4G systems, but with a higher flexibility thanks to PHY layer numerology. More in details, a PRB is composed by 12 subcarriers whose spacing is $\Delta f = 15 \times 2^{\mu}$ KHz with $\mu = 0, 1, 2$ on seven OFDM symbols, whose duration changes accordingly to the subcarrier spacing [27]. The set of PRBs is indicated with $\mathcal{N}$, whose dimension is N_{RB} and depends on the considered frequency band.

In an actual system the user data rate request must be converted to an integer number of PRB to be assigned. As detailed later, we propose a two-layer resource allocation scheme, where PRBs assignment is performed in two

TABLE 1. RAN slicing approaches comparison.

Approach	ref.	Solution	Slice Isolation	Slice Priority	user QoS	Resource Shortage
Single-Layer	[9]	Stochastic Game	NO	NO	user queue length	NO
	[10]	Heuristic - iterative	Min satisfaction level	slices weights	NO	NO
	[11]	Heuristic - iterative	Min PRB	NO	NO	NO
	[12]	Barrier method	Min PRB	slices weights	NO	NO
	[13]	Fisher markets	NO	sharing level	NO	NO
	[14]	Off-line reinforcement and heuristic	NO	NO	YES	prob. outage
	[15]	Semi-online heuristic	NO	sharing level	NO	NO
Two-Layer	[7], [17]	Inter-slice:fixed allocation Inter-slice: customized	dedicated resources	NO	NO	NO
	[18]	Inter-slice: heuristic Intra-slice: PF	NO	NO	NO	NO
	[19]	Inter-slice: fixed allocation or PF Intra-slice: not specified	fixed sharing	NO	not specified	NO
	[20]	Inter-slice: heuristic - iterative Intra-slice: not specified	Min PRB/capacity	NO	not specified	Admission control
	[21]	Inter-slice: auction game Intra-slice: not specified	Min PRB	YES	NO	NO
	[22]	Intra and Inter-slice: stackelberg game	NO	NO	YES	NO
	[23]	Intra and Inter-slice: PF	NO	YES	NO	NO
	[24]	Intra-slice: bankruptcy game Inter-slice: not specified	NO	NO	slice typical rate	YES

phases that work with different time-scales and channel state information.

At system level, in order to define how many PRBs have to be assigned to each slice, k-th user is characterized by its averaged signal to noise ratio (SNR), $\bar{\Gamma}_k$ with $k = 1 \cdots K$, and consequently, by an approximated transmission rate per PRB, that is $\bar{r}_k = Blog(1 + \bar{\Gamma}_k)$ where B represents the bandwidth of a PRB. Differently, at slice level the channel diversity in each PRB is considered, hence, the k-th user is characterized by the SNR value experienced on the n-th PRB that is $\Gamma_{k,n}$, and consequently the data rate on the n-th PRB is $r_{k,n} = Blog(1 + \Gamma_{k,n})$.

A. SLICE CONFIGURATION

Here we consider three main 5G service classes (i.e., $S = 3$): eMBB, URLLC, mMTC. Each slice is characterized by a priority p_s (with $\sum_{\mathcal{S}} p_s = 1$) , and by a number of requested PRBs that depends on the users' QoS requirements:

- k-th eMBB user, with $k \in \mathcal{K}_1$, is characterized by a data rate request R_k that can be transformed into the average number of PRBs requested by the k-th user as $n_k = \left\lceil \frac{R_k}{\bar{r}_k} \right\rceil$.
- k-th URLLC user, with $k \in \mathcal{K}_2$, has to transmit a packet with length P bits and is characterized by a transmission delay budget δ_k. As a consequence, the k-th user asks for an averaged number of PRB[1] $n_k = \left\lceil \frac{P}{\delta_k \bar{r}_k} \right\rceil$.
- k-th mMTC user, with $k \in \mathcal{K}_3$, transmits with a fixed low data rate R_{mMTC}, that results in a single PRB per user, that is $n_k = 1 \; \forall k \in \mathcal{K}_3$.

As a consequence each slice needs a total amount of resources $N_s = \sum_{k \in \mathcal{K}_s} n_k$ to satisfy all its users' requests.

[1]The delay value is only referred to the transmission time, queuing delay as well as retransmission time are not considered.

We are interested in the case of resource shortage, that is $\sum_{s \in \mathcal{S}} N_s > N_{RB}$, indeed, in the opposite case the solution is trivial, each slice (and each user) has assigned the exact amount of requested resources.

IV. PROBLEM FORMULATION AND SOLUTION

The aim of this paper is to find a solution to the problem of allocating resources to different slices taking simultaneously into account their priority, a sufficient level of isolation among slices, fairness and users QoS requirements. Being previous characteristics highly variable, we aim at a flexible solution based on a weighted fairness among slices, where weights can be changed depending on the scenario. Moreover, to have an affordable computational complexity, and for guaranteeing slices' customization, we want first divide resources among slices (*iter-slice* scheduler), and then allocate them within each slice in a customized way (*intra-slice* scheduler). In particular, the inter-slice scheduler determines how many PRBs to assign to each slice, and partitions the set of PRBs in S subsets, $\mathcal{N}_s$ with $s = 1, \cdots, S$. Then each intra-slice scheduler independently allocates the PRBs within its subset to its users. The inter-slice scheduler works on a frame-basis, with the knowledge of the total amount of resources requested by each slice, while the intra-slice scheduler works at transmission time interval (TTI) level taking into account users' instantaneous SNR values.

Hence, we want to solve successively two sub-problems:

- *Sub-problem 1*

$$\begin{aligned} \max_{\hat{\mathbf{N}}} & \left\{ \sum_{s \in \mathcal{S}} p_s (N_s)^n log(\hat{N}_s) \right\} \\ \text{s.t.:} & \sum_{s \in \mathcal{S}} \hat{N}_s \leq N_{RB} \\ & \hat{N}_s \leq N_s \quad \forall s \in \mathcal{S} \end{aligned} \tag{1}$$

- *Sub-problem 2*

$$\begin{aligned}
&\max_{\mathbf{A}} \left\{ \sum_{s\in\mathcal{S}} w_s U_s \right\} \\
&\text{s.t.:} \sum_{n\in\mathcal{N}_s} \sum_{k\in\mathcal{K}_s} a(k,n) \le \hat{N}_s \\
&\quad \sum_{n\in\mathcal{N}_s} a(k,n) r_{k,n} - \min_{\substack{n\in\mathcal{N}_s \\ a(k,n)=1}} r_{k,n} < R_k \quad \forall k \in \mathcal{K} \\
&\quad \sum_{s\in\mathcal{S}} \sum_{k\in\mathcal{K}_s} a(k,n) \le 1 \quad \forall n = 1,\cdots,N_{RB} \qquad (2)
\end{aligned}$$

In (1), $\hat{N}_s$ is the total amount of resources assigned to the s-th slice regardless to which users will be assigned, and $\hat{\mathbf{N}}$ is the vector of length S whose elements are $\hat{N}_s$. Moreover, the term $(N_s)^n$ is introduced to manage different levels of isolation among slices, in particular $n \in [0, 1]$ and N_s takes into account the load of a slice. This means that when $n = 0$ the slice load is not considered, the only element that impacts on the utility is the priority of the slices. As a consequence, variations of the slices' load do not affect the resource assignment (slices' isolation). Conversely, when $n \neq 0$, the utility varies also with the slice load. In particular, the traffic load influence increases with n, and when $n = 1$ the isolation effect is completely lost. The *log* function guarantees fairness among slices.

In (2), $\mathbf{A}$ is the allocation matrix whose element $a(i,j)$ is *one* if the j-th resource block is assigned to the i-th user, *zero* otherwise. The term U_s represents the utility of the s-th slice that can be differently defined for each slice, and w_s is a normalization factor. In particular, we consider here the α-utility [13]

$$U_s = \sum_{k\in\mathcal{K}_s} \begin{cases} \dfrac{\left(\sum_{n\in\mathcal{N}} a(k,n) r_{k,n}\right)^{1-\alpha}}{1-\alpha} & \text{if} \quad \alpha \neq 1 \\ log\left(\sum_{n\in\mathcal{N}} a(k,n) r_{k,n}\right) & \text{if} \quad \alpha = 1 \end{cases} \qquad (3)$$

where α is a parameter that can be selected in order to have different scheduling policies: $\alpha = 1$ - *Proportional Fair* (PF), $\alpha = 2$ - *Delay Fairness* (DF), $\alpha \to$ inf - *Max-Min* and $\alpha = 0$ - *Sum Rate*. We want to guarantee fairness within slices. As a consequence we consider the following scheduling approaches: PF for eMBB and mMTC slices and DF for URLLC slice. This is only an example of utility that can be used at slice level, in fact intra-slice scheduler can be customized depending on the slices's preferences. The main focus of the paper is on the inter-layer scheduling and how it interacts with the slices.

In both sub-problems, (1) and (2), the first constraint imposes that the total number of resources cannot overcome the available ones (i.e., N_{RB} for the whole system and $\hat{N}_s$ for each slice), and the second that assigned resources cannot overcome the requested ones. In particular, in (2) since the users' rate requests are continuous values while the assigned resources are discrete values, the assigned rate can overcome the requested one. However, the second constraint assures that only the needed PRBs are assigned, in fact removing the PRB with the lowest rate among those assigned to the user, the rate request is no longer satisfied. Moreover, third constraint in (2) assures that one PRB can be assigned to only one user.

A. INTER-SLICE SCHEDULER

First we want to solve the problem (1), splitting resources among slices in a fair way in accordance with their priority, and with a sufficient level of isolation among slices. The problem (1) is integer, and to solve it we relax the constraint on the value that can be assumed by the elements of $\hat{\mathbf{N}}$. We assume that $\hat{N}_s$ is a real value. Being the problem constrained by inequalities the solution can be found by extending the method of Lagrange multipliers to the Karush Kuhn Tucker (KKT) conditions. Constraints are linear functions, hence the solution space is a convex set and the optimization function is concave, thus KKT conditions are sufficient to find an optimum. The inequality conditions multiplied by a factor μ_i are added to the cost function in a similar way of the equality in the method of Lagrange multipliers. The expression for the optimization problem becomes

$$\begin{aligned}
\max_{\hat{\mathbf{N}}} \mathcal{L}(\hat{\mathbf{N}}_s, \mu_0, \cdots, \mu_S) &= \max_{\hat{\mathbf{N}}} \sum_{s\in\mathcal{S}} p_s N_s^n log(\hat{N}_s) \\
&\quad - \mu_0 \left(\sum_{s\in\mathcal{S}} \hat{N}_s - N_{RB} \right) - \sum_{s\in\mathcal{S}} \mu_s \left(\hat{N}_s - N_s \right) \qquad (4)
\end{aligned}$$

where $\mathcal{L}(\hat{\mathbf{N}}_s, \mu_0, \cdots, \mu_S)$ is the Lagrangian that depends on $\hat{\mathbf{N}}$ and the multipliers μ_s, with $s = 0, \cdots, S$.

KKT conditions are:

- *Stationarity*:

$$\nabla_{\hat{\mathbf{N}}_s} \mathcal{L}(\hat{\mathbf{N}}_s, \mu_0, \cdots, \mu_S) = \left(\frac{\partial \mathcal{L}}{\partial \hat{N}_1}, \cdots, \frac{\partial \mathcal{L}}{\partial \hat{N}_S} \right) = 0$$

  that can be rewritten as

$$\begin{cases} \dfrac{p_1 N_1^n}{\hat{N}_1} - \mu_0 - \mu_1 = 0 \\ \vdots \\ \dfrac{p_S N_S^n}{\hat{N}_S} - \mu_0 - \mu_S = 0 \end{cases}$$

- *Slackness conditions*

$$\begin{cases} \mu_0 \left(\sum_{s\in\mathcal{S}} \hat{N}_s - N_{RB} \right) = 0, & \mu_0 \ge 0 \\ \mu_s \left(\hat{N}_s - N_s \right) = 0, & \mu_s \ge 0 \quad \forall s = 1, \cdots, S \end{cases}$$

that can be rewritten as

$$\begin{cases} \mu_0 = 0, \quad \sum_{s\in\mathcal{S}} \hat{N}_s < N_{RB} & \| \quad \mu_0 > 0, \quad \sum_{s\in\mathcal{S}} \hat{N}_s = N_{RB} \\ \mu_1 = 0, \quad \hat{N}_1 < N_1 & \| \quad \mu_1 > 0, \quad \hat{N}_1 = N_1 \\ \vdots \\ \mu_S = 0, \quad \hat{N}_S < N_S & \| \quad \mu_S > 0, \quad \hat{N}_S = N_S \end{cases}$$

As stated before we are interested in finding the solution when $\sum_{s\in\mathcal{S}} N_s > N_{RB}$, that is, the available resources are not sufficient for satisfying whole slices' requests. The opposite condition can be solved with a trivial solution, each slice achieves exactly the amount of required PRBs, and the exceeding PRBs are let unused: $\hat{N}_s = N_s$. As a consequence, the condition $\mu_0 = 0$, implying $\sum_{s\in\mathcal{S}} \hat{N}_s < N_{RB}$, is not considered.

Conversely, when $\mu_0 > 0$ in general we have: $\mu_i > 0$ with $i = 1, \cdots, R$, $\mu_l = 0$ with $l = R+1, \cdots, S$ and $0 \leq R \leq S$. This leads to

$$\max_{\hat{\mathbf{N}}} \mathcal{L}(\hat{\mathbf{N}}_s, \mu_0, \cdots, \mu_R) = \sum_{s\in\mathcal{S}} p_s N_s^n log(\hat{N}_s) - \mu_0 \left(\sum_{s\in\mathcal{S}} \hat{N}_s - N_{RB} \right) - \sum_{i=1}^{R} \mu_i \left(\hat{N}_i - N_i \right) \quad (5)$$

hence, we have:

$$\begin{cases} \frac{p_i N_i^n}{\hat{N}_i} = \mu_0 + \mu_i & i = 1, \cdots, R \\ \frac{p_l N_l^n}{\hat{N}_l} = \mu_0 & l = R+1, \cdots, S \\ \sum_{s\in\mathcal{S}} \hat{N}_s = N_{RB} \\ \hat{N}_i = N_i & i = 1, \cdots, R \end{cases}$$

whose solution is:

$$\begin{cases} \hat{N}_i = N_i & i = 1, \cdots, R \\ \hat{N}_i = \dfrac{p_i N_i^n \left(N_{RB} - \sum_{l=1}^{R} N_l \right)}{\sum_{w=R+1}^{S} p_w N_w^n} & i = R+1, \cdots, S \end{cases} \quad (6)$$

where the value of R can be derived as described in the Appendix.

In particular, in the two extreme cases $R = 0$ and $R = S$ we have:

1) $R = S \rightarrow \hat{N}_s = N_s \quad \forall s$
 if $\sum_{s\in\mathcal{S}} N_s = N_{RB}$;
2) $R = 0 \rightarrow \hat{N}_s = \frac{p_s N_s^n}{\sum_{l\in\mathcal{S}} p_l N_l^n} N_{RB} \quad \forall s$
 if $\frac{p_s N_s^n}{\sum_{l\in\mathcal{S}} p_l N_l^n} N_{RB} < N_s \quad \forall s$;

This means that resources are allocated to the slices proportionally to the term $p_s N_s^n$ that takes into account priority and total traffic of the slice. Depending on n, we can have different approaches: with $n = 0$ radio resources are assigned only depending on the slices' priority, but independently on the slice traffic load, this allows the maximum "*isolation*" among slices. With $n = 1$ radio resources are assigned proportionally to the traffic, hence, a variation on traffic load (i.e., number of users or service profile) on a slice has effects also on the others, thus "*isolation*" is not guaranteed. Choosing different values of $n \in [1, 0]$ we can have different trade-offs between the two extreme approaches.

Actually, to solve the problem we have assumed $\hat{N}_s$ is a real value, but the number of PRBs that can be allocated is integer, hence, the final solution is obtained by means a rounding operation. First we assign to each slice a number of PRBs that is given by the next smaller integer of $\hat{N}_s$. Then if $\sum_{s\in\mathcal{S}} \lfloor \hat{N}_s \rfloor < N_{RB}$, the remaining PRBs are assigned according to a RR approach to the slices. In this way we avoid assigning more than N_{RB} PRBs.

B. INTRA-SLICE SCHEDULER

Once the number of resources per slice has been determined by the inter-slice scheduler (i.e., $\hat{N}_s$), each slice has assigned a subset $\mathcal{N}_s$ of PRBs and performs users scheduling adopting its own policy. Hence, as stated before the utility U_s in (2) can be differently defined for each slice.

Due to the complexity of the problem, we propose an iterative solution that takes into account the different slices' policies. As stated before, many other different intra-layer scheduling algorithms can be used without affecting the validity of the overall proposed architecture and the slices management policy. The most important thing at slice level is the customization that is allowed by this kind of scheduling architecture.

The allocation matrix $\mathbf{A}^{(0)}$ is initially empty.

Then PRBs are distributed among users with the goal of maximizing the utility function (2). We define a *gain-factor* that expresses the advantage of assigning a PRB to a user. In particular, assuming that at the i-th iteration the matrix allocation is $\mathbf{A}^{(i)}$, and the k-th user has assigned a data rate equal to $\hat{R}_k^{(i)} = \sum_{n\in\mathcal{N}_s} a(k, n)^{(i)} r_{k,n}$, the *gain-factor* at the iteration $(i+1)$ is defined as

$$g_{k,s}^{(i+1)} = \begin{cases} w_s \left[log(\hat{R}_k^{(i)} + r_{k,\hat{n}_k}) - log(\hat{R}_k^{(i)}) \right] & \text{if} \quad s = 1, 3 \\ w_s \left[\dfrac{1}{\hat{R}_k^{(i)}} - \dfrac{1}{\hat{R}_k^{(i)} + r_{k,\hat{n}_k}} \right] & \text{if} \quad s = 2 \end{cases} \quad (7)$$

The index $\hat{n}_k$ is used to indicate the available (i.e., not yet assigned) PRB $\in \mathcal{N}_s$, where the k-th user experiences better channel conditions. Hence, the user $\hat{k}$-th, that has the maximum value of $g_{k,s}^{(i+1)}$ is selected among the users for which

- the data rate request has not been already satisfied: $\hat{R}_k^i < R_k$;
- the resources assigned to the user's slice are lower than the value determined by the inter-slice scheduler: $\sum_{n\in\mathcal{N}_s} \sum_{k\in\mathcal{K}_s} a(n, k)^{(i)} < \hat{N}_s$.

Finally, the PRB is assigned to the selected user: $a(\hat{k}, \hat{n}_{\hat{k}})^{(i+1)} = 1$. If one or more users have the same minimum value of the gain-factor, one of them is randomly selected. Then the next iteration is performed.

Iterations stop when all PRBs have been assigned or all the requests have been satisfied.

C. COMPUTATIONAL COMPLEXITY

The proposed approach presents low complexity and for this reason represents a suitable solution for practical implementation. In particular, the two-layer approach decouples the slices and users allocation problem, thus reducing the complexity of a single multi-objective scheduler that has to manage a huge number of users with different requirements and constraints. Moreover, the two schedulers use two different time-scales, thus allowing to each slice to manage its users with flexible TTIs as specified by the 5G standard.

In particular:

- *Inter-slice scheduler* - it needs to know only the number of PRBs requested by each slice that is calculated on a large-time scale (for example on a frame basis). For a given configuration (i.e., p_s and n) the number of PRBs per slice $\hat{N}_s$ is simply calculated as in (6), where only the term $p_s N_s^n$ must be calculated for each slice. The complexity is substantially an elementary operation per slice, hence, is linear with S (i.e., $\mathcal{O}(S)$).
- *Intra-slice scheduler* - it works at TTI level taking into account instantaneous SNR values. Given the value of $\hat{N}_s$ derived by the inter-slice scheduler, each slice has to allocate the PRBs to its K_s users. The procedure is iterative, and at each iteration an utility gain is calculated for each user that has not already received the requested rate. In the worst case at every iteration all users must be considered (actually, users that have already received the required resources are not considered in the successive iterations). Therefore, the complexity of one iteration for the s-th slice is $K_s \times C_s$, where C_s is a constant representing the complexity for computing the utility gain expressed in (7) that is made of few simple operations. The number of iterations in the worst case is equal to the number of PRBs assigned to the slice $\hat{N}_s$, hence, the complexity of the intra-slice scheduler for the s-th slice is $\mathcal{O}(\hat{N}_s \times K_s)$.

V. BENCHMARK APPROACHES

The behavior of the proposed approach is evaluated in comparison with different benchmark methods.

1) First of all we can observe that the proposed approach when $n = 0$ corresponds to the case in which radio resources are pre-assigned to different slices. In particular, the resources partitioning among the slices depends only on the priority,[2] and not on the traffic load. However, the optimal solution avoids that one slice has an overabundance of resources respect to its traffic load, while the others have a lack of resources. As specified in Appendix, if a slice has an overabundance of resources due to its high priority, it receives only the requested ones, while the remaining are let for other slices. Hence, the approach is similar to that used in [7].
2) Moreover, we consider the method proposed in [24] because it is based on users' QoS requests in terms of PRBs and assumes lack of resources to satisfy all users' requests as our method. However, differently from our method, all users of a slice request the same typical rate, while here we assume random distributed values. The solution in [24] is found by means a bankruptcy game as briefly described later.
3) Finally, we consider the method proposed in [15]. Similarly, to the method proposed here, in [15] the authors aim at defining a weighted fairness policy, where each user has assigned a fraction of resources that is proportional to the sharing level over the number of users of its slice. This solution does not consider users' requirements and PRB assignment, hence, we have adapted it to our context as detailed later.

We have to underline that the first two benchmark approaches ([7], [24]) are based on a two-layer RAN slicing architecture as ours, but focus only on the inter-slice scheduler. For what concerns the intra-layer scheduler we have adopted the same solution described in IV-B. Differently, the solution proposed in [15] is a single layer scheduler, hence resources are directly assigned to the end-users.

A. BANKRUPTCY GAME ALLOCATION STRATEGY

The approach proposed in [24] is based on the bankruptcy game that is a special form of cooperative game. The inter-slice scheduler and the slices are the bankrupt company and debtors in the game, respectively. The game consists of a finite set of debtors (players) $\mathcal{S}$ and a characteristic function v that maps coalitions of players to real numbers. More in detail, if C is a coalition of players, then $v(C)$ provides the total expected sum of payoffs the members of C, that is [24]

$$v(C) = max \left\{ 0, N_{RB} - \sum_{\substack{s \in \mathcal{S} \\ s \notin C}} N_s \right\} \tag{8}$$

Players can cooperate forming independent coalitions to capture more benefit, and members of coalitions share benefits according to their contribute to coalitions. The number of possible coalitions in $\mathcal{S}$ is 2^S. The optimal cost-sharing solution is provided by the Shapley value that is designed to allocate collectively gained benefit between players in a fair way. The Shapley value is computed as:

$$\Phi_i(v) = \sum_{C \in \mathcal{S} \setminus i} \left(\frac{|C|!(S - |C| - 1)!}{S!} v(C \cup i) - v(C) \right) \tag{9}$$

that is the sum over all possible coalitions C of $\mathcal{S}$ not containing the i-th slice. The characteristic function is calculated for all the possible coalitions and then the Shapley value of each player (slice) is calculated and rounded to determine the

[2]It could be based also on different criteria.

amount of PRBs assigned to each slice. It is important to point out that this method does not take into account slices priority.

B. INVERSE LOAD PROPORTIONAL FAIRNESS

The method proposed in [15] allows resource sharing among different tenants following a weighted proportional fair criterion. Each tenant represents a network slice that is characterized by a sharing level, similar to the priority defined here. Radio resources are dynamically assigned to users, proportionally to the sharing level of the slice they belong to, over the number of users of that slice. Authors show that this allocation is Pareto optimal. Differently from our approach, this is based on a single-layer scheduler and considers the capacity maximization regardless the actual users' QoS requirements. Moreover, the authors do not consider an actual physical resource discretization (i.e., PRBs) but assume that the resources' fraction that is assigned to a user can assume any value. We have adapted this solution to our context, considering the PRB assignment and the users' requests. As a consequence the k-th user belonging to the s-th slice has assigned an amount of resources that is given by:

$$\hat{n}_k = N_{RB}\frac{\frac{p_s}{K_s}}{\sum_{j\in\mathcal{S}}\frac{p_j}{K_j}} \tag{10}$$

However, if the resources assigned to a user are higher than those requested by the user (i.e., $\hat{n}_k > n_k$), we perform a redistribution among users to avoid waste of bandwidth. The radio resources allocated to a user are inversely proportional to the number of users of its slice, hence, users of heavy loaded slices receive less resources.

VI. NUMERICAL RESULTS

This section presents the numerical results derived to validate the effectiveness of the proposed method. We have assumed that the data-rate requests of the eMBB slice are exponentially distributed with mean value R_{eMBB}, while the delay of the URLLC slice users is uniformly distributed on a given interval $[\delta_{min} - \delta_{max}]$ whose mean value is δ_{av}. Consequently, the normalization factors used in (2) are $w_1 = \frac{1}{log(R_{eMBB})}$, $w_s = \frac{\delta_{av}}{P}$ and $w_3 = \frac{1}{log(R_{mMTC})}$.

Main simulation parameters are summarized in Tab. 2.

TABLE 2. Simulation parameters.

Parameter	Value
Considered Area	$0.5\ Km^2$
BS EIRP	$46dB$
N_{PRB} in $10ms$ frame	500
R_{eMBB}	$[1-10]Mbps$
$[\delta_{min} - \delta_{max}]$	$[1-2]ms$
P	100 bit
R_{mMTC}	$10kbps$

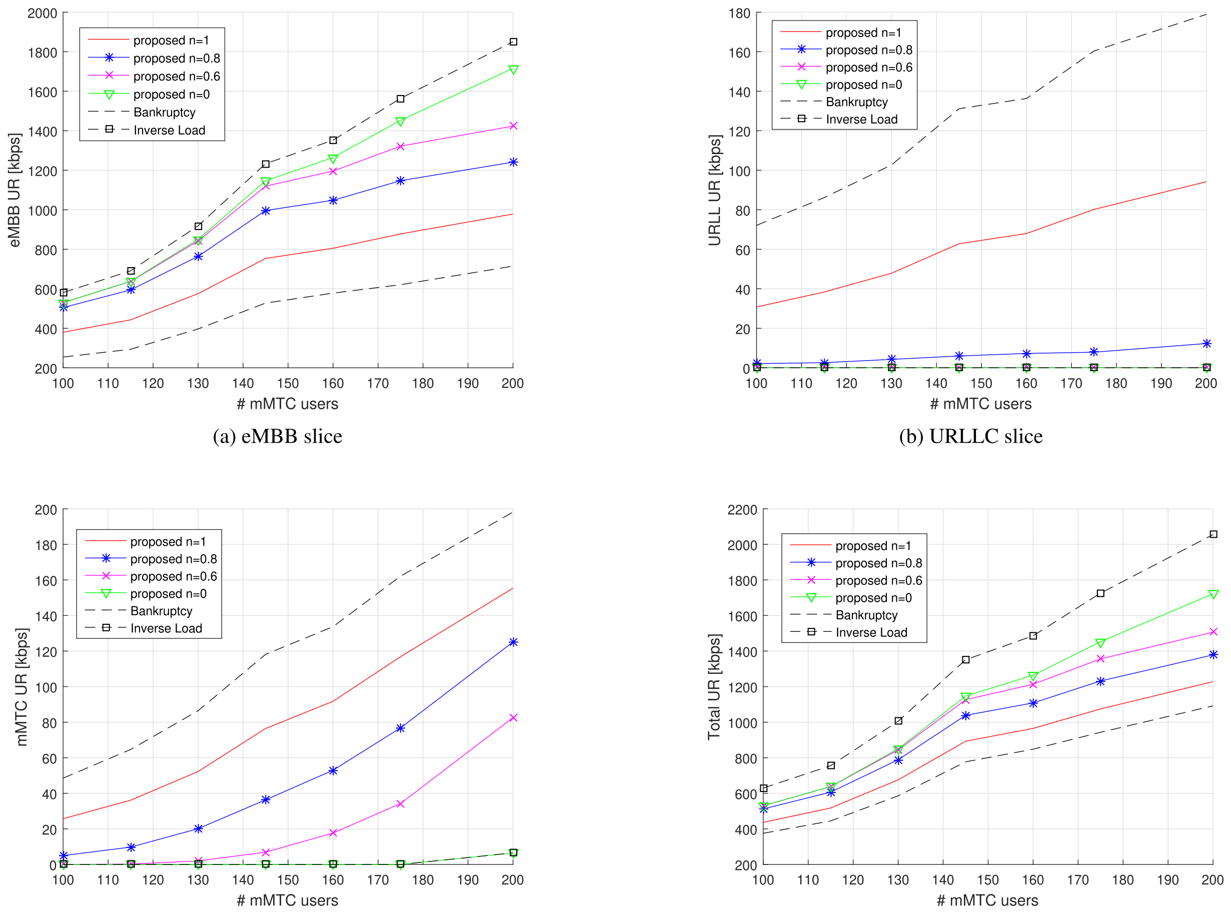
Moreover, we have assumed here the Hata-cost model in Urban scenario as pathloss model with a log normal distributed shadowing.

Simulation outputs of several realizations have been averaged to make the results independent on the particular distribution of users and data traffic.

First of all we want to show the flexibility of the proposed approach in terms of isolation. Fig. 1 shows the total amount of *unsatisfied data rate requests*, that is defined as the sum of the difference between the requested and the achieved data rate (i.e., $(n_k - \hat{n}_k)r_k$) of each user. In particular, the subfigures show the unsatisfied requests (URs) for the three slices and for the whole system, respectively. These results have been derived by assuming a variable mean number of mMTC active devices while $K_1 = 10$, and $K_2 = 30$. The mean data rate for eMBB devices is $R_{eMBB} = 1Mbps$, while to derive the UR of the mMTC slice we have assumed an average data rate request of $10kbps$ for mMTC devices, and all slices have the same priority. In all figures we can note that the performance of the proposed approach varies with n, but different slices present different behaviors depending on their traffic load. When $n = 0$ resources are assigned to the slices taking into account their priority and not their traffic load. The solution allows to avoid waste of precious resources, in fact each slice receives at maximum the requested PRBs, hence, eventual *extra*-bandwidth is assigned to the slices that need it.

Increasing the value of n the isolation among slices decreases. As a consequence, the traffic overload of a slice is spread to all slices.

eMBB slice is the one with the highest load due to the high data rate requests. Therefore, as we can see in Fig. 1a, decreasing n, eMBB URs increase because increasing isolation the traffic overload mainly affects the slice that caused the overload (i.e., eMBB itself in this case). As we can expect, the other two slices present the opposite behavior with n: increasing the isolation ($n \to 0$) these are less affected by the traffic overload of the eMBB slice, and have sufficient resources for their own traffic. In particular, URLLC slice presents the lowest traffic among the three slices, hence, as we can see in (Fig.1c), it is sufficient a low level of isolation to have a significant improvement of the UR performance, in particular with $n = 0.6$ we have that all users' requests are satisfied. Moreover, when n decreases curves tend to become flat, indeed a variation of the number of mMTC devices has less impact. For what concerns the third slice (Fig.1c), its traffic increases with the number of mMTC devices, and as a consequence, also URs increase. Indeed, we can note that even with $n = 0$, if the mean number of devices is higher than 166 (that is $N_{RB}/3$), the slice has not dedicated resources for its users. It is important to underline, that even with complete isolation ($n = 0$) and constant eMBB traffic, the performance of the eMBB slice varies because the amount of resources let unused by the other two slices changes. In fact, URLLC and mMTC (when the number of mMTC devices is lower than $N_{RB}/3$) slices let some resources to the eMBB slice, because they have assigned only the requested ones. However, increasing the number of mMTC devices, the resources surplus of the mMTC slice decreases, and hence, the amount of *extra* resources received by the



(a) eMBB slice (b) URLLC slice (c) mMTC slice (d) System

FIGURE 1. **Unsatisfied data rate Requests (UR) vs K_3 ($K_1 = 10$, $K_2 = 30$, $R_{eMBB} = 1Mbps$ and $p_1 = p_2 = p_3$).**

eMBB slice decreases as well. Moreover, when the number of mMTC devices overcomes $N_{RB}/3$, also the mMTC slice receives part of the surplus of the URLLC resources, thus the extra bandwidth for eMBB slice further reduces and its URs increase.

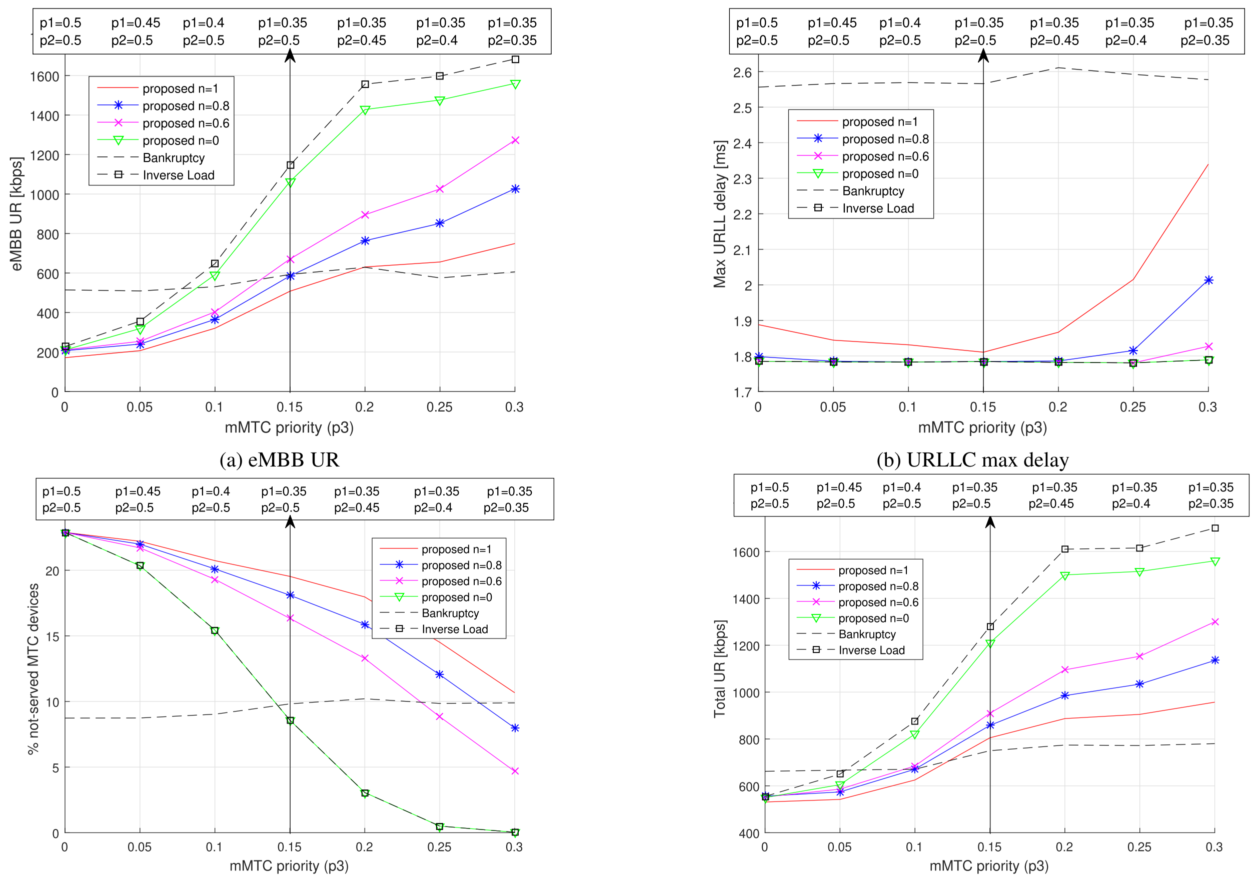
Being URs of the first slice significantly higher than the others, the total amount of URs (Fig. 1d) follows the behavior of the highly loaded slice (eMBB).

In all these figures we can note that the effect of the parameter n is more evident when the traffic load of the mMTC slice increases (i.e., a higher number of mMTC users), and becomes more comparable to the eMBB traffic load.[3] Similarly, being the URLLC and eMBB traffic loads significantly unbalanced, URLLC performance only slightly varies with n, with the exception of total absence of isolation ($n = 1$).

For what concerns the benchmarks we can note that *Bankruptcy* approach presents better performance at system level, close to the case $n = 1$ (i.e., proportional fair), but presents no isolation among slices. Indeed, the Bankruptcy approach tends to give more resources to the slice with higher traffic. Conversely, the method proposed in [15], named "*Inverse Load*" (i.e., the amount of resources assigned to a user is inversely proportional to the number of users in the slice), achieves the highest isolation because the amount of resources assigned to each slice substantially depends only on its priority, while the number of users in each slice affects only the amount of resources that is assigned to each user within a slice. Hence, while URLLC and mMTC slices have performance equal to the proposed method with $n = 0$, for what concerns the eMBB slice, the performance worsens.

The results presented in Fig. 1 show the flexibility of the proposed scheme, and how it is possible to achieve the most suitable behavior simply modifying the isolation parameter. In fact, performance of the proposed method can move from the complete isolation (similarly to the Inverse Load method) to the maximum multiplexing gain (similarly to the Bankruptcy approach). Moreover, as shown later, the proposed method allows to manage the slice priority, that provides another degree of freedom to the inter-slice resource allocation.

[3]Having the eMBB slice the highest traffic load, this slice mainly influences performance.



(a) eMBB UR (b) URLLC max delay

(c) % not-served MTC devices (d) System UR

FIGURE 2. Slices' performance vs mMTC priority p_3 ($\bar{K}_1 = 10, \bar{K}_2 = 50, \bar{K}_3 = 150$ and $R_{eMBB} = 1Mbps$).

The choice of the parameter n, as well as the priority, depends on SLAs and the operative context. Isolation should be increased when the risk of a reduction of the allocated resources due to events external to the slice is not accepted because the services must be guaranteed (e.g. a public safety slice in an emergency situation or high-value services slice). Obviously, isolation reduces the multiplexing gain, thus resulting in a lower efficiency of the network, and likely a higher cost for the slices' owners. Conversely, the isolation factor can be reduced if a certain fluctuation of the assigned resources is accepted with the benefit of reduced costs. Moreover, the lack of isolation can be partially compensated with the priority values, for example a URLLC slice can accept some of fluctuations of the assigned resources, but should receive a precedence in the resource usage to guarantee the delay requirement to its users. In fact, the proposed scheme is flexible not only in terms of isolation, but also in terms of slices' priority management. Figs. 2 show the performance of the three slices and the system when priorities vary.

Differently from previous figures, here the amount of URs is converted into a slice's specific metric, in order to show the effects of resource allocation on the QoS requirement that characterizes the slice, as explained in Sect.III. In particular, the figures represent the total URs of the eMBB slice, the maximum delay of the URLLC slice and the percentage of not-served devices of the mMTC slice, respectively. Performance is given as a function of the priority of the mMTC slice (p_3), and different combinations of the other two priorities. More in detail, starting from the point, $p_3 = 0$, and $p_1 = p_2 = 0.5$, p_3 is increased by reducing the priority of the first slice up to $p_1 = 0.35$, while $p_2 = 0.5$ remains constant. Then, in the second part of the figures, $p_1 = 0.35$ remains constant, while the p_3 is increased by decreasing the priority of the second slice up to $p_2 = 0.35$. To better understand the results, it is important to stress that in this scenario the URLLC slice has limited traffic load, this means that with $n = 0$ it has a assigned only the requested resources, while the excess is assigned to the other two slices. When p_2 decreases in favor of p_3, URLLC slice gives up some resources to the mMTC slice. On the other side, having a high load the mMTC slice uses all the assigned resources (this means that increasing p_3 at the expense of p_2 the amount of extra-resources that is assigned to the eMBB slice tends to zero). Fig. 3a shows that eMBB URs increase as p_3 increases. In fact, initially the increase of p_3 occurs at the expense of p_1 that has to let go some resources. Successively, even if p_1 is

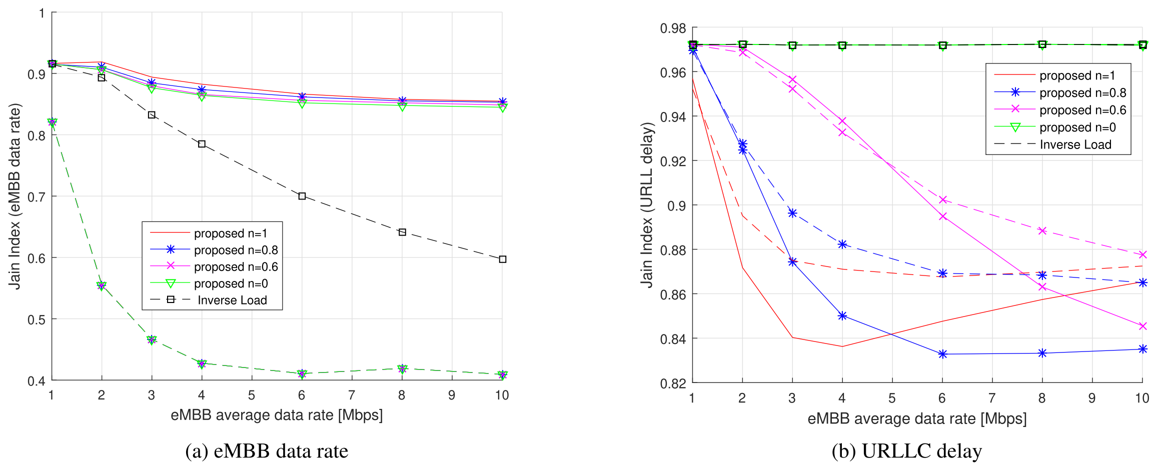


FIGURE 3. **Jain index eMBB and URLLC slices vs R_{eMBB} ($K_1 = 10$, $K_2 = 20$, $K_3 = 50$ and $p_1 = p_2 = p_3$). Solid lines represent PF while dashed DF.**

constant, reducing the priority of the second slice the surplus of URLLC resources that are redistributed to the eMBB slice decreases as well. This is particularly evident when $n = 0$. As a consequence when the surplus of URLLC resources to be redistributed tends to zero, the eMBB URs curve for $n = 0$ tends to become flat (starting approximately from $p_2 = 0.45$). This is because if there is not *extra*-resources redistribution the eMBB slice status does not change. In the second slice, Fig. 3b, we can note that even if its priority is constant (first part of the figure) the maximum delay decreases because the priority of the highest loaded slice (eMBB) is decreasing, hence, it has lower impact on it. In the second part of the figure the delay increases because the priority of the URLLC slice is decreasing. We can note that the delay highly suffers the influence of other slices' traffic, hence, there is need of sufficient isolation. With $n = 0.6$ the delay requirement (i.e., maximum delay under $2ms$) is met even with low values of priority. For what concerns the mMTC slice (Fig. 2c), obviously increasing its priority, the amount of not-served machines decreases. The decreasing is more rapid if there is higher isolation (n close to 0). When $p_3 = 0$ there is no difference varying n because in this case the mMTC slice always receives what the other slices left unused, and this does not depend on the isolation factor. As in the previous set of figures, the curves of the total amount of URs (Fig. 2d) follow the behavior of the highly loaded slice (eMBB). In all cases, we can note that the Bankruptcy approach has constant performance, because priority is not considered. While, as observed before, the Inverse Load approach, follows the proposed method with $n = 0$, but has worst performance for the slice with the highest load (eMBB).

Finally, we show the different behaviors of the *intra-slice* scheduler depending on the selected option (i.e., the parameter α defined in IV). In particular, different kinds of *fairness* among the users of a slice can be selected. Figs. 3a and 3b show the *Jain index* of the achieved data-rate for the eMBB slice and of delay for the URLLC slice, respectively. Two different intra-slice allocation policies are considered: PF ($\alpha = 1$) and DF ($\alpha = 2$), and are compared with the fairness of the "Inverse Load" method that considers only the number of users and not their requests. The Jain index provides a measure of the fairness of a given performance x and is defined as $\frac{(\sum_{i=1}^{N} x_i)^2}{N \sum_{i=1}^{N} x_i^2}$. In the eMBB slice we can note, that the fairness in terms of achieved data rate is significantly higher using PF rather than DF. Indeed, DF presents a low Jain index value, that decreases if the system load increases and tends to 0.4, while the PF Jain Index tends to 0.85. For low values of R_{eMBB} almost all the users' requests are satisfied, hence, differences between the two approaches tend to reduce. Conversely, DF approach allows higher fairness (in terms of delay) among the users in URLLC slice. In this case, we can note different behaviors of the proposed method when n varies, because the traffic variations in the eMBB slice differently affect the URLLC slice. In particular, for $n = 0$, there is no difference between DF and PF because URLLC slice has sufficient resources to satisfy all its users' requests thanks to the isolation. Differences between the two methods are more evident in the eMBB slice where the amount of requested resources is higher, and a different allocation policy has higher impacts on the performance. Results are presented for $p_1 = p_2 = p_3$, but different priority values do not change these conclusions. Indeed, priority affects only the inter-slice scheduler. Obviously, the priority (as well as n) changes the amount of resources assigned to each slice, and hence, can determine lack or surplus of resources thus leading to a more or less evident gain of one approach respect to the other as shown in the figures. Hence, the intra-slice scheduler is not directly affected by the values of n and the priority, but obviously these parameters (with others as the N_{RB}, the slices' traffic load, the propagation conditions) change the number of resources assigned to a slice, hence change the operative conditions in which the scheduler works. The benchmark method does not introduce fairness in terms of data rate, indeed allocation is performed considering only the number of users. In the URLLC slice, where the number of resources is sufficient to accommodate all the users' requests, the behavior of the Inverse Load method is the same of the proposed one with $n = 0$. Conversely, fairness among eMBB users significantly worsens.

VII. CONCLUSIONS

The paper focused on Radio Access Network (RAN) slicing. In particular, in order to manage the scarce radio resources, RAN slicing requires dynamic resource management policies able to take into account different issues, such as efficiency, isolation and customization, that can be also in contrast each other. Toward this goal, this paper proposed a two-layer scheduler for an efficient, flexible and low complexity RAN slicing approach suitable for application in actual systems. The results showed that the proposed approach allows to achieve a suitable trade-off among different aspects that vary with the operative context. In particular, the inter-layer scheduler is able to achieve the desired trade-off between multiplexing gain, isolation and priority simply changing few parameters in the utility function. This provides a high flexibility, and hence, a single algorithm can be used in different contexts. When isolation requirements is dominant, each slice receives a bandwidth that is proportional to its priority, even if unused resources of a slice are redistributed among the other slices that need them. When the multiplexing gain is the most important requirement the isolation can be reduced, and consequently, the traffic load increasing in one slice has a detrimental effect also on other slices. This can be partially compensated with the priority, that in any case allows to give a certain precedence to a slice in the resource assignment. Moreover, thanks to the two-layer architecture, the intra-layer scheduler can be customized, not only in terms of resource allocation policy but also in terms of scheduling-time following the 5G requirements. We have shown that using different intra-slice scheduler policies allows to better satisfy the specific QoS requirements of a slice. We have focused here on fairness among users of a slice, but many other approaches could be considered at slice level (intra-slice scheduler) depending on the specific needs, without affecting the validity of the overall proposed system and the slices management policy. The most important thing at slice level is the customization that is allowed by this kind of scheduling architecture.

APPENDIX

As seen in Sect. IV-A, the optimal solution to the first sub-problem is

$$\begin{cases} \hat{N}_i = N_i & i = 1, \cdots, R \\ \hat{N}_i = \dfrac{p_i N_i^n \left(N_{RB} - \displaystyle\sum_{l=1}^{R} N_l \right)}{\displaystyle\sum_{w=R+1}^{S} p_w N_w^n} & i = R+1, \cdots, S \end{cases}$$

To determine the value of R an iterative procedure can be applied. The goal is to satisfy the second constraint in (1), hence, R must be determined so that no slices have assigned more resources than those required.

Initially $R^{(0)} = 0$ and $\hat{N}_s^{(0)} = \left(\frac{p_s N_s^n}{\sum_{i=1}^{S} p_i N_i^n} N_{RB} \right)$.

Assuming that there are g^0 slices that have $\hat{N}_s^{(0)} > N_s$ with $s = 1, \cdots, g^0$ (for simplicity and without loss of generality, we assume these are the first g^0 slices), at the first iteration we have $R^{(1)} = g^0$, $\hat{N}_i = N_i$ for $i = 1, \cdots, R^{(1)}$ and $\hat{N}_s^{(1)} = \frac{p_s N_s^n \left(N_{RB} - \sum_{l=1}^{R^{(1)}} N_l \right)}{\sum_{w=R^{(1)}+1}^{S} p_w N_w^n}$ $s = R^{(1)} + 1, \cdots, S$. Similarly, at the next iteration if g^1 slices have $\hat{N}_s^{(1)} > N_s$ with $s = g^0 + 1, \cdots, g^1$ we have $R^{(2)} = g^0 + g^1$, $\hat{N}_i = N_i$ for $i = 1, \cdots, R^{(2)}$ and $\hat{N}_s^{(2)} = \frac{p_s N_s^n \left(N_{RB} - \sum_{l=1}^{R^{(2)}} N_l \right)}{\sum_{w=R^{(2)}+1}^{S} p_w N_w^n}$ $s = R^{(2)} + 1, \cdots, S$.

This procedure is repeated until the t-th iteration when there are not more slices that have $\hat{N}_s^{(t)} > N_s$ (i.e., $g^t = 0$). Hence, $R = R^{(t)}$.

Substantially, the procedure looks for the value of R that allows to give to the slices that have high priority only the amount of required resources. The excess-resources that would be assigned for the priority and not for the effective traffic load, are instead let to be used by the other slices.

REFERENCES

[1] M. Maternia, S. E. El Ayoubi, M. Fallgren, P. Spapis, Y. Qi, D. Martín-Sacristán, Ó. Carrasco, M. Fresia, M. Payaró, M. Schubert, J. S. Bedo, and V. Kulkarni. (Apr. 10, 2016). *5G PPP Use Cases and Performance Evaluation Models*. [Online]. Available: http://www.5g-ppp.eu/

[2] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94–100, May 2017.

[3] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz, and H. Bakker, "Network slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 72–79, May 2017.

[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.*, vol. 20, no. 3, pp. 2429–2453, 3rd Quart., 2018.

[5] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "How should I slice my network? A multi-service empirical evaluation of resource sharing efficiency," in *Proc. ACM MobiCom*, New Delhi, India, Nov. 2018, pp. 191–206.

[6] *TS 23.251 Technical Specification Group Services and System Aspects; Network Sharing; Architecture and Functional Description (Release 10)*. document 3GPP, V10.6.0, Jun. 2013.

[7] A. Ksentini and N. Nikaein, "Toward enforcing network slicing on RAN: Flexibility and resources abstraction," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 102–108, Jun. 2017.

[8] R. Ferrus, O. Sallent, J. P. Romero, and R. Agusti, "On 5G radio access network slicing: Radio interface protocol features and configuration," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 184–192, May 2018.

[9] F. Fu and U. C. Kozat, "Wireless network virtualization as a sequential auction game," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.

[10] M. Kalil, A. Shami, and Y. Ye, "Wireless resources virtualization in LTE systems," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr./May 2014, pp. 363–368.

[11] M. I. Kamel, L. B. Le, and A. Girard, "LTE wireless network virtualization: Dynamic slicing via flexible scheduling," in *Proc. IEEE Veh. Tech. Conf. (VTC)*, Sep. 2014, pp. 1–5.

[12] J. He and W. Song, "AppRAN: Application-oriented radio access network sharing in mobile networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 3788–3794.
[13] P. Caballero, A. Banchs, G. de Veciana, and X. Costa-Pérez, "Network slicing games: Enabling customization in multi-tenant networks," in *Proc. IEEE INFOCOM*, May 2017, pp. 1–9.
[14] H. D. R. Albonda and J. Pérez-Romero, "An efficient RAN slicing strategy for a heterogeneous network with eMBB and V2X services," *IEEE Access*, vol. 7, pp. 44771–44782, 2019.
[15] P. Caballero, A. Banchs, G. de Veciana, X. Costa-Pérez, "Multi-tenant radio access network slicing: Statistical multiplexing of spatial loads," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3044–3058, Oct. 2017.
[16] C.-Y. Chang and N. Nikaein, "Ran runtime slicing system for flexible and dynamic service execution environment," *IEEE Access*, vol. 6, pp. 34018–34042, 2018.
[17] D. Marabissi and R. Fantacci, "Heterogeneous public safety network architecture based on ran slicing," *IEEE Access*, vol. 5, pp. 24668–24677, 2017.
[18] M. Hu, Y. Chang, Y. Sun, and H. Li, "Dynamic slicing and scheduling for wireless network virtualization in downlink lte system," in *Proc. 19th Int. Symp. Wireless Pers. Multimedia Commun. (WPMC)*, Nov. 2016, pp. 153–158.
[19] Y. Zaki, L. Zhao, C. Goerg, and A. Timm-Giel, "LTE wireless virtualization and spectrum management," in *Proc. WMNC*, Oct. 2010, pp. 1–6.
[20] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan, "NVS: A substrate for virtualizing wireless resources in cellular networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1333–1346, Oct. 2012.
[21] M. Jiang, M. Condoluci, and T. Mahmoodi, "Network slicing in 5G: An auction-based model," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.
[22] J. Hu, Z. Zheng, B. Di, and L. Song, "Tri-level Stackelberg game for resource allocation in radio access network slicing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.
[23] Z. Jian, W. Muqing, M. Ruiqiang, and W. Xiusheng, "Dynamic resource sharing scheme across network slicing for multi-tenant c-rans," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC Workshops)*, Aug. 2018, pp. 172–177.
[24] Y. Jia, H. Tian, S. Fan, P. Zhao, and K. Zhao, "Bankruptcy game based resource allocation algorithm for 5G Cloud-RAN slicing," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6.
[25] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia, and A. Banchs, "Mobile traffic forecasting for maximizing 5G network slicing resource utilization," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.
[26] T. Guo and R. Arnott, "Active LTE RAN sharing with partial resource reservation," in *Proc. IEEE 78th Veh. Technol. Conf. (VTC Fall)*, Sep. 2013, pp. 1–5.
[27] A. A. Zaidi, R. Baldemair, V. Moles-Cases, N. He, K. Werner, and A. Cedergren, "OFDM numerology design for 5G new radio to support IoT, eMBB, and MBSFN," *IEEE Commun. Standards Mag.*, vol. 2, no. 2, pp. 78–83, Jun. 2018.

DANIA MARABISSI (SM'13) received the Ph.D. degree in informatics and telecommunications engineering from the University of Florence, in 2004, where she works as an Assistant Professor. She was the winner of the contest Futuro in Ricerca 2013 funded by the Italian Ministry of Education and is responsible for the scientific activities of the Technologies for Information and Communication (TICom) Consortium, a joint venture between the University of Florence and Leonardo Company SpA. She has been involved in several national and European research projects and has authored several technical papers published in international journals and conferences and patents. Her current research interests are physical and access layers issues. She has been involved in the organization of several international conferences as a symposium/workshop chair. She is an Associate Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, *Transaction on Emerging Telecommunication Technologies*, *Future Internet* and *The Scientific World Journal*.

ROMANO FANTACCI (F'05) received the Ph.D. degree in telecommunications from the University of Florence, Italy, in 1987, where he is currently a Full Professor with the Department of Information Engineering. He has been involved as responsible in several national and international research projects and is the author of more than 300 papers published in prestigious communication science journals. He was funder of the Information Communication Technology Consortium (TICom), a joint venture between the University of Florence and Leonardo Company and has been appointed as the President, since 2010. He guests edited special issues in the IEEE Journals and magazines and served as a symposium chair of several IEEE conferences. He has an active role in the IEEE and ComSoc from several years. He received several Best paper Award at the IEEE COMSOC Conferences, the AEI/IEEE Young Resercah Renato Mariani Award, in 1982, the IEE IERE Benefactor premium in 1990, the IEEE COMSOC Award Distinguished Contributions to Satellite Communication, in 2002 and the IEEE COMSOC Award Distinguished Contributions to Wireless Communication, in 2015.

He was an Associate Editor of several prestigious journals and he is currently serving as a Regional Editor if the *IET Communications* and an Associate Editor of the *International Journal of Communication Systems*.