CARES: Computation-Aware Scheduling in Virtualized Radio Access Networks

Dario Bega, Albert Banchs, *Senior Member, IEEE*, Marco Gramaglia, Xavier Costa-Pérez, *Member, IEEE*, and Peter Rost, *Senior Member, IEEE*

***Abstract*— In a virtualized radio access network (RAN), baseband processing is performed by software running in cloud-computing platforms. However, current protocol stacks were not designed to run in this kind of environment; the high variability on the computational resources consumed by RAN functions may lead to eventual computational outages (where frames are not decoded on time), severely degrading the resulting performance. In this paper, we address this issue by re-designing two key functions of the protocol stack: 1) *scheduling*, to select the transmission of those frames that do not result in computational outages, and 2) *modulation and coding scheme (MCS) selection*, to downgrade the selected MCS in case no sufficient computational resources are available. We formulate the resulting problem as a joint optimization and compute the (asymptotically) optimal solution to this problem. We further show that this solution involves solving an NP-hard problem, and propose an algorithm to obtain an approximate solution that is computationally efficient while providing bounded performance over the optimal. We thoroughly evaluate the proposed approach via simulation, showing that it can provide savings as high as 80% of the computational resources while paying a small price in performance.**

***Index Terms*— 5G, computation-aware scheduling, virtualized RAN, joint scheduling, MCS selection.**

I. Introduction

FOLLOWING current trends in *network virtualization*, virtualized RAN (Radio Access Network) is considered as one of the key components of future 5G networks [1]. Enabled by the Network Function Virtualization (NFV) paradigm, this technology virtualizes the RAN functions and moves them out from the base stations. The RAPs (Radio Access Points) only perform simple processing of the Radio Frequency (RF) signal while the majority of the baseband processing is executed in Virtual Network Functions (VNFs) running in cloud-computing platforms.[1] The cloud-computing platforms typically run the baseband processing of a number of RAPs in a single location; depending on the scenario, such location may be placed in edge nodes close to the antennas (e.g., in MEC – multi-access edge computing) or in the network core (e.g., in centralized RAN).

There is wide consensus in the scientific and industrial communities on the advantages of RAN virtualization. Indeed, this technology provides, among other benefits, high flexibility and increased multiplexing gains. Driven by these benefits, substantial research and standardization efforts have been devoted over the last few years to the design and analysis of virtualized RAN solutions [2]–[6].

In spite of the efforts conducted to date, there are still some unresolved research challenges with RAN virtualization; one of them is the dimensioning of the computational resources. The dimensioning of resources should be based on the processing demands of the different functions; however, the computational load of functions such as baseband processing is known to be highly variable [3], which makes the resource allocation based on peak demands highly inefficient. A much more efficient strategy would be to pool the baseband processing and allocate resources to this pool based on typical (rather than peak) demands [7].[2] This implies that occasionally the baseband processing pool may not have all the resources needed to process all frames within the involved timing constraints; following [10], hereafter this is referred to as a *computational outage*.

The current RAN protocol stack has been designed under the assumption that required computational resources are always available, and RAN functions are not prepared to cope with computational outages [11]. Indeed, when such computational outages occur (e.g., when the decoding function cannot meet the given deadline), current virtualized RAN implementations [12] just drop the frame being processed, and as a result they see their performance severely degraded.

Manuscript received March 28, 2018; revised September 13, 2018; accepted September 17, 2018. Date of publication October 11, 2018; date of current version December 10, 2018. This work was supported in part by the H2020 5G-MoNArch Project, University Carlos III of Madrid, under Grant 761445, and in part by NEC Europe Ltd., through the 5G-Transformer Project under Grant 761536. The associate editor coordinating the review of this paper and approving it for publication was K. Kansanen. *(Corresponding author: Dario Bega.)*

D. Bega and A. Banchs are with the IMDEA Networks Institute, 28918 Leganés, Spain, and also with the Telematic Engineering Department, University Carlos III of Madrid, 28911 Leganés, Spain (e-mail: dario.bega@imdea.org; banchs@it.uc3m.es).

M. Gramaglia is with the Telematic Engineering Department, University Carlos III of Madrid, 28911 Leganés, Spain (e-mail: mgramagl@it.uc3m.es).

X. Costa-Pérez is with NEC Laboratories Europe, 69115 Heidelberg, Germany (e-mail: xavier.costa@neclab.eu).

P. Rost is with Nokia Bell Labs, 81541 Munich, Germany (e-mail: peter.m.rost@nokia-bell-labs.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2018.2873324

[1] This is the case, for instance, with options 7 and 8 for the functional split defined by 3GPP [2].

[2] Indeed, one would expect that resources are typically allocated dynamically based on estimates of the average demand. However, such predictions may fail due to the fluctuations of the demand or other events such as, e.g., an unexpected *flash-crowd*. While in such cases more resources may be provided by adding Virtual Machines or containers to the cloud, this usually happens at time scales (of tens of seconds or minutes) [8] that are much larger than the user QoS requirements (of hundreds of milliseconds) [9].

In order to provide a more robust behavior against computational outages, a re-design of the RAN protocol stack is required. As a first step towards this end, in this paper we focus on the design of two key functions of the protocol stack: *scheduling* and *modulation and coding scheme (MCS) selection*. By making these functions *computationally aware*, our solution provides a graceful performance degradation in presence of computational outages, in contrast to the severe degradation with the current stack. Our two key ideas are as follows:

- By being aware of the frames' computational resource consumption when *scheduling* transmissions, we can select those frames that prevent the system from incurring into computational outages.
- As shown in [13], computational resource consumption decreases when selecting more robust *modulation-and-coding schemes* (MCS); hence, by reducing the selected MCS for some frames we can control the overall computational resource consumption.

We remark that the above techniques are not decoupled, and they may be combined in different ways to mitigate the impact of computational outages. Following this, the key contribution of this paper is the design of a mechanism that *jointly* optimizes scheduling and MCS selection to improve the performance of a virtualized RAN system with (temporarily) *limited* computational resources.

A. Related Work

There is a vast amount of work in the literature focusing on the design of algorithms for scheduling and for modulation-and-coding scheme selection. However, most of this work assumes unlimited amount of computational resources, and focuses only on communications resources (namely, channel quality). In this context, many scheduling algorithms have been designed to optimize performance under the given channel quality (see, e.g., [14]–[16]). From all these algorithms, the most relevant ones are those aiming at providing proportional fairness [17]–[22], which is the criterion we focus on in this paper. However, these algorithms do not take into account computational resources, and the choice of the optimal schedule may substantially vary when taking them into account.

There have also been many proposals on modulation and coding scheme (MCS) selection algorithms [23]–[25]. However, all these algorithms assume that the only constraint when selecting the MCS is the channel quality, and do not consider any other aspect. Our approach is therefore fundamentally different, as channel quality only provides us with an upper bound on the MCS that can be selected, but depending on the available computational resources we may select lower MCSs.

In traditional networks, scheduling and MCS selection are fully decoupled: first, the highest MCS under the current channel quality is selected for each device, and then devices are scheduled with the selected MCS for each of them. This is fundamentally different in our case, where we can counter computational resources outages either by scheduling less demanding users or by selecting a more robust MCS for some of them, and hence we need to address scheduling and MCS selection *jointly*.

Current trends on virtualization and cloudification of network functions have led to some research work focusing on understanding the computational load incurred by RAN functions [26]. Bhaumik *et al.* [27] analyze this based on a real experimental testbed, but do not consider the impact of the SNR, which is known to influence computational load very substantially. Rost *et al.* [13] provide a more accurate model for computational load. In this paper we employ the model in [13], which is the most accurate proposed to date, for our performance evaluation experiments.

Perhaps the closest work to ours is that of [10]. In that work, the authors propose to downgrade the MCS of scheduled users to fight computational outages. However, in contrast to our approach, [10] does not consider radio resource scheduling. In the performance evaluation, we show that our approach (which jointly optimizes scheduling and MCS selection) substantially outperforms an approach that combines a standard scheduling with the MCS selection of [10] (relying on two separate optimizations).

The re-design of the VNF internals to make them robust to computational outages will be a fundamental cornerstone of virtualized mobile network protocol stacks. By building network functions that are *elastic* with respect to the computational resources available in the cloud, infrastructure providers can host more tenants on the same infrastructure, having a better resources utilization and increasing thus the CPU cycle per dollar ratio.

B. Key Contributions

The key contributions of the paper are as follows. After presenting the system model (Section II), we introduce the maximum step algorithm, which maximizes the resulting performance at each step, and show that this algorithm is asymptotically optimal (Section III). As the maximum step algorithm involves solving an NP-hard problem, we devise the Computation-AwaRE Scheduling (CARES) algorithm, which is computationally efficient and provides bounded performance guarantees (Section IV). Finally, we exhaustively evaluate the performance of the proposed approach via simulation, showing that CARES achieves savings in terms of computational capacity of up to 80% while paying a very small price in performance (Section V).

II. Model and Problem Formulation

In this section, we present our system model and formulate the optimization problem that will drive the computational-aware scheduling of the users. The model is inspired in LTE-A but is generally applicable to any cellular system.

A. System Model

We consider a typical cloud-computing platform that is serving a set $\mathcal{S}$ of RAPs. The amount of resources devoted to the baseband processing for these RAPs is given by the computational capacity C. Each RAP has a number B of resource blocks. We denote by $\mathcal{U}_s$ the set of users associated with RAP $s \in \mathcal{S}$.

In line with standard LTE schedulers [28]–[32], time is divided into Transmission Time Intervals (TTIs) and at every TTI scheduling decisions are performed for the next TTI based on the current channel quality of the different users. Such a scheme allows to take into account the SNR level of each user resulting from fading, interference and other effects. We can thus select users when they experience their best channel quality; this is commonly referred to as "multi-user diversity", as it exploits the fact that different users will experience peaks in their channel quality at different times. More specifically, at each TTI a central scheduler decides (i) the mapping of users to resource blocks, and (ii) the MCS selected for each user. This decision is taken with the goal of improving the overall system performance given the SNR of each user and the available computational capacity C.

In our system model, we assume that each user sees the same SNR level in all resource blocks at a given TTI. The SNR level constrains the set of MSC schemes that may be selected for a user: we denote the set of possible MCS schemes available for user u at TTI t by $\mathcal{M}_u(t)$. Specifically, we may select an MCS scheme from this set under the constraint that all the resource blocks assigned to the user in a TTI have to employ the same MCS (as imposed by LTE-A). We further let $c_{u,m}(t)$ denote the computational load incurred to decode a resource block for user u under MCS $m \in \mathcal{M}_u$ at TTI t (given the user's channel at that time). Following [10], we impose that the aggregate computational load in the system (given by the individual loads $c_{u,m}(t)$) cannot exceed the computational capacity C.

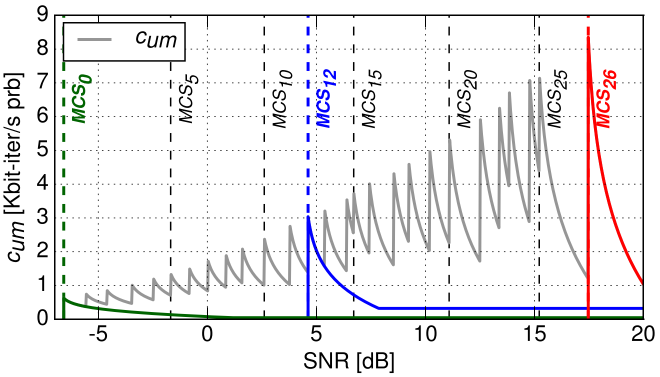

Fig. 1. Computational load a function of SNR level, obtained from the analysis of [13].

Note that the computational load $c_{u,m}(t)$ incurred by user u at time t is a function of the SNR level at that time and the MCS selected for the user. Indeed, the computational effort required to decode a sub-frame increases with the number of iterations, the number of code blocks and the number of information bits, among other factors; in turn, the number of iterations depends on the SNR, and the number of code blocks and information bits depend on the selected MCS. To determine the value of $c_{u,m}(t)$, we follow the model provided in [13], according to which the computational load for MCS m with a given SNR is equal to

$$c_m(\text{SNR}) = r_m \max\left[1, \frac{\log_2\left(-\frac{(\zeta-2)\log_{10}\hat{\epsilon}_{\text{channel}}}{\mathcal{K}\zeta}\right)}{\log_2(\zeta-1)} - \frac{2\log_2\left[\log_2(1+\text{SNR})-r_m\right]}{\log_2(\zeta-1)}\right]$$

where r_m is the rate provided by one resource block with MCS m, $\hat{\epsilon}_{\text{channel}}$ is the constraint on the channel outage probability, and ζ and $\mathcal{K}$ are constant parameters; following [13], we set $\zeta = 6$ and $\mathcal{K} = 0.2$ and $\hat{\epsilon}_{\text{channel}} = 0.1\%$. Moreover, in the above expression we set a minimum computational load which corresponds to one iteration in order to account for large SNR values. Fig. 1 illustrates the resulting computational load per resource block, $c_{u,m}$, as a function of the instantaneous SNR level; specifically, the figure shows (i) the computational load of a few selected MCSs (in different colors) and (ii) the computational load of the highest possible MCS for each SNR value (in gray).

Note that the above computational load values correspond to a cloud environment (e.g., a datacenter) consisting of commodity hardware, which is the natural scenario for a virtualized RAN system such as the one considered in this paper. If we used hardware with accelerators such as GPU (Graphics Processing Unit) or FPGA (Field-Programmable Gate Array), the computational load would change as a result of parallelizing some of the tasks. In any case, the algorithm we propose can be applied to any hardware platform, simply by taking the corresponding values for the computational load and capacity.

B. Legacy Proportional Fair (PF) Scheduler

In this paper, we aim at providing proportional fairness (PF), which is a widely accepted criterion for sharing resources. Indeed, much effort in the literature has been devoted to the design of PF scheduling algorithms (see, e.g., [17]–[22]). In the following, we present the formulation of a legacy PF scheduler under no computational capacity constraints, which will be taken as a baseline for our approach.

Let $x_{u,b}(t)$ be a variable that indicates if resource block b is assigned to user u at TTI t. Let $r_{u,b}(t)$ be the rate provided to user u by resource block b at that time, depending on the user's SNR. A scheduling algorithm involves deciding the assignment of resource blocks to users (i.e., the $x_{u,b}(t)$ values) depending on the users' channel quality at each point in time (reflected by $r_{u,b}(t)$). Once a scheduling decision is taken, the rate received by user u at TTI t is equal to $r_u(t) = \sum_{b \in \mathcal{B}_{s(u)}} x_{u,m}(t) r_{u,b}(t)$, where $\mathcal{B}_{s(u)}$ is the set of resource blocks at user u's RAP. The average rate of user u is then given by the following expression, which corresponds to an exponential moving average updated at each TTI [17], [19], [21], [22], [33]:

$$R_u(t+1) = (1-\alpha)R_u(t) + \alpha r_u(t),$$

where $r_u(t)$ is the rate received by user u at TTI t and α is a smoothing factor which discounts past values of $r_u(t)$ to reflect that more recent transmissions provide a higher value to the user. As the PF criterion aims to maximize the long-term throughputs [17]–[22], this implies that, with this criterion, recent transmissions are not valued significantly higher than previous ones. Following this, in the rest of the paper we assume that the discount factor α takes a small value.

With the above, PF scheduling can be formulated as an optimization problem that maximizes the overall performance or *utility*, given by the sum of the logarithms of the average rates, under the constraint that a resource block cannot be assigned to more than one user:

$$\begin{aligned}&\underset{\{x_{u,m}(t)\}}{\text{maximize}} \sum_{u\in\mathcal{U}} \log \mathbb{E}[R_u(t)]\\ &\text{subject to} \sum_{u\in\mathcal{U}} x_{u,b}(t) \le 1, \quad x_{u,b}(t)\in\{0,1\}\ \forall u,b,t.\end{aligned}$$

The above problem has been solved by the literature, showing that performance is optimized when at a given TTI t, the user with the highest $r_u(t)/R_u(t)$ is scheduled (see, e.g., [22]).

C. Computation-Aware Proportional Fair Scheduling and MCS Selection

The previous problem formulation assumes that computational resources are unlimited; under such conditions, optimizing MCS selection and scheduling are decoupled problems: users first select the highest MCS allowed by their radio conditions, and then scheduling chooses the users that maximize the system's utility given the pre-selected MCSs. In this paper, we focus on computational-constrained systems where scheduling and MCS selection decisions take computational resources into account, which couples these decisions and leads to a joint optimization problem. In the resulting problem formulation, scheduling and MCS selection are driven by the decision variable $x_{u,b,m}(t)$, which indicates that resource block b is assigned to user u with MCS m at TTI t. Based on this variable, the rate received by user u at TTI t is given by:

$$r_u(t) = \sum_{b\in\mathcal{B}_{s(u)}} \sum_{m\in\mathcal{M}_u(t)} x_{u,b,m}(t) r_{u,m}(t), \tag{1}$$

where $\mathcal{B}_{s(u)}$ is the set of resource blocks at user u's RAP, $\mathcal{M}_u(t)$ is the set of MCS options available for user u at time t (according to the radio conditions at that time) and $r_{u,m}(t)$ is the rate provided to user u by one resource block with MCS m. With this, the goal of maximizing overall performance or *utility* in terms of PF in this system leads to the following optimization[3]:

$$\underset{\{x_{u,b,m}(t)\}}{\text{maximize}} \sum_{u\in\mathcal{U}} \log \mathbb{E}[R_u(t)] \tag{2a}$$

$$\text{subject to} \sum_{u\in\mathcal{U}}\sum_{b\in\mathcal{B}_{s(u)}} \sum_{m\in\mathcal{M}_u(t)} x_{u,b,m}(t)\, c_{u,m}(t) \le C, \quad \forall t, \tag{2b}$$

$$\sum_{u\in\mathcal{U}} \sum_{m\in\mathcal{M}_u(t)} x_{u,b,m}(t) \le 1, \quad \forall b,t, \tag{2c}$$

$$\begin{aligned}&x_{u,b,m}(t) + x_{u,b',m'}(t) \le 1,\\ &\quad \forall u,b,b',m' \neq m,t,\end{aligned} \tag{2d}$$

$$x_{u,b,m}(t) = 0, \quad \forall u,b,m \notin \mathcal{M}_u(t), \tag{2e}$$

$$x_{u,b,m}(t) \in \{0,1\} \ \ \forall u,b,m,t. \tag{2f}$$

The above formulation corresponds to a joint optimization problem involving both scheduling and MCS selection, under the constraints that the consumption of computational resources does not exceed the available capacity (2b), each resource block is only assigned once (2c), a user can employ only one MCS[4] (2d) and we cannot select a higher MCS than the one allowed by current radio conditions (2e). The optimization problem is similar to the legacy proportional fair scheduling presented in Section II-B, except for the constraint on computational load. However, this constraint makes the problem much harder, as it couples scheduling decisions with MCS selection (which are decoupled in the legacy case) and makes the problem NP-hard (in contrast to legacy PF scheduling).

The above problem formulation allows to select an MCS lower than the allowed by the user's radio conditions (which we refer to as an MCS *downgrade*) when computational resources are limited. While this may have a negative impact on throughput, our optimization guarantees that such MCS downgrades will only be performed when this contributes to improving the overall system performance. Indeed, the results presented in Section V show that such selective MCS downgrades can drastically reduce computational resource consumption while paying a minimum price in terms of throughput performance.

The underlying assumption behind the above problem formulation is that all users are backlogged, i.e., they always have a frame ready for transmission, and the algorithm devised hereafter is based on this assumption. However, in case some of the users are not backlogged, we can adapt our algorithm following the approach proposed in [34] to ensure that (*i*) a user cannot steal resources from the other users upon becoming backlogged, and (*ii*) under low loads, i.e., when there are no backlogged users, the system provides low latencies (see [34] for more details).

III. Optimal Solution

We next present the *maximum step algorithm* and show that this algorithm is asymptotically optimal, meaning that it provides optimal performance when $\alpha \to 0$. Note that, as mentioned before, we can expect that α will be set to a small value, which means that in practice the *maximum step algorithm* will provide a very good approximation to the optimal solution.

The *maximum step algorithm* maximizes the objective function increase at each step, i.e., it maximizes the following

[3] Our formulation focuses on uplink scheduling, as the load at the cloud-computing platform is dominated by uplink frames [27]; however, the formulation could be easily extended to incorporate downlink scheduling by simply adding downlink transmissions with their (low) computational cost.

[4] In particular, following the LTE-A specifications mentioned in Section II-A, this constraint forces that if x_{ubm} is equal to 1 for a certain user u and a certain MCS m, it cannot be equal to 1 for the same user u and another MCS m'.

expression for each t,

$$\sum_{u\in\mathcal{U}} \log R_u(t+1) - \sum_{u\in\mathcal{U}} \log R_u(t).$$

The above can be expressed as follows:

$$\sum_{u\in\mathcal{U}} \log R_u(t+1) - \sum_{u\in\mathcal{U}} \log R_u(t) = \sum_{u\in\mathcal{U}} \log \frac{R_u(t+1)}{R_u(t)}$$
$$= \sum_{u\in\mathcal{U}} \log \frac{\alpha \sum\limits_{b\in\mathcal{B}_{s(u)}} \sum\limits_{m\in\mathcal{M}_u(t)} x_{u,b,m}(t) r_{u,m}(t) + (1-\alpha) R_u(t)}{R_u(t)},$$

where $s(u)$ is the RAP serving user u.

Maximizing the above expression is equivalent to maximizing

$$\prod_{u\in\mathcal{U}} \left(\frac{\alpha \sum_{b\in\mathcal{B}_{s(u)}} \sum_{m\in\mathcal{M}_u(t)} x_{u,b,m}(t) r_{u,m}(t)}{R_u(t)} + (1-\alpha) \right).$$

If we divide the above by $(1-\alpha)^{|\mathcal{U}|}$, the maximization still holds, yielding to

$$\prod_{u\in\mathcal{U}} \left(\frac{\alpha \sum_{b\in\mathcal{B}_{s(u)}} \sum_{m\in\mathcal{M}_u(t)} x_{u,b,m}(t) r_{u,m}(t)}{(1-\alpha) R_u(t)} + 1 \right).$$

To simplify notation, let us denote the number of resource blocks assigned to user u with MCS m at time t by $n_{u,m}(t)$, i.e., $n_{u,m}(t) = \sum_{b\in\mathcal{B}_{s(u)}} x_{u,b,m}(t)$, which allows to write the above as a more compact expression: $\prod_{u\in\mathcal{U}} \left(\frac{\alpha \sum_{m\in\mathcal{M}_u(t)} n_{u,m}(t) r_{u,m}(t)}{(1-\alpha) R_u(t)} + 1 \right)$. It can be seen that this expression is equivalent to

$$\prod_{u\in\mathcal{U}} \prod_{m\in\mathcal{M}_u(t)} \left(\frac{\alpha n_{u,m}(t) r_{u,m}(t)}{(1-\alpha) R_u(t)} + 1 \right),$$

since the terms for which $n_{u,m}(t) = 0$ have no effect in either of the expressions.

Let us denote by $\Delta(t)$ the strategy followed for the assignment of users to resource blocks as well as for selecting the MCS of each user. The definition of a strategy is determined by variables $\{n_{u,m}(t), \forall u, m\}$. According to the analysis presented in this section, we have that the strategy followed by the maximum step algorithm is given by

$$\arg\max_{\Delta(t)} \prod_{u\in\mathcal{U}} \prod_{m\in\mathcal{M}_u(t)} \left(\frac{\alpha n_{u,m}(t) r_{u,m}(t)}{(1-\alpha) R_u(t)} + 1 \right) \quad (3)$$

under the constraints (2b)-(2f).

Hence, the maximum step algorithm takes the scheduling decision at each TTI t that maximizes (3) given the value of the various variables at that time. More specifically, it decides the number of resource blocks to assign to each user, $n_{u,m}(t)$, depending on the rate a user would receive, $r_{u,m}(t)$, and her current average rate based on previous transmissions, $R_u(t)$.

The following theorem shows that this algorithm is asymptotically optimal as $\alpha \rightarrow 0$, which (as discussed in Section II-B) is a reasonable assumption when dealing with long-term rates. The theorem also assumes that the channel state of the various RAPs can be modeled as a finite Markov chain. Note that this is a mild constraint, as Markov chains allow for very general models. Indeed, many works rely on similar assumptions to model wireless channels [33], [35]–[39]. The proof of the theorem is provided in Appendix I.

Theorem 1: If the wireless channels of the RAPs behave according to a Markov chain, then the maximum step algorithm becomes (asymptotically) optimal as $\alpha \rightarrow 0$.

While the maximum step algorithm provides (asymptotically) optimal performance, it involves finding the best strategy $\Delta(t)$ at each t, which is very costly in terms of computational complexity. This is confirmed by the following theorem, which shows that the maximum step algorithm is an NP-hard problem and thus infeasible in practical scenarios. The proof of the theorem is provided in Appendix II.

Theorem 2: The maximum step algorithm is NP-hard.

IV. Approximate Solution: CARES Algorithm

In this section we propose an heuristic, the *Computational-AwaRE Scheduling* algorithm (CARES), to obtain an approximate solution to the optimization problem of (2a)-(2f). The proposed approach (i) incurs an acceptable computational overhead, and (ii) provides a performance that is provably close to the optimal by a constant factor.

The CARES algorithm proposed here is inspired by the algorithm proposed in [40] for the multiple-choice knapsack problem. While the problem addressed in [40] and CARES have some similarities, there are also fundamental differences: (i) in [40] there are no constraints on the number of items that can be inserted in a knapsack, while in CARES this is limited by the number of available resource blocks; (ii) in CARES, a single user cannot employ different MCSs, while [40] does not have such a constraint, (iii) in CARES, a user can be assigned several resource blocks, while items cannot be repeated in [40], and (iv) CARES aims to utilize all resource blocks, in contrast to [40]. These differences have strong implications in terms of (i) the algorithm design (the branching module has been adapted to the specifics of our problem, while the improvement module is completely new), (ii) the theoretical analysis (the complexity analysis differs substantially and the performance bound proof has been adapted, adjusting Lemma 1 and incorporating Lemma 2) and (iii) the performance results (see Figure 5 for a comparison of CARES against a variation of [40] in terms of network utilization).

A. Algorithm Description

The CARES algorithm builds on the following approximation:

$$\log \left(\frac{\alpha n_{u,m}(t) r_{u,m}(t)}{(1-\alpha) R_u(t)} + 1 \right)$$
$$\approx n_{u,m}(t) \log \left(\frac{\alpha r_{u,m}(t)}{(1-\alpha) R_u(t)} + 1 \right), \quad (4)$$

which is accurate for small α; indeed, when α is small both the left-hand side and the right-hand side of the above equation can be approximated by

$$\frac{\alpha n_{u,m}(t) r_{u,m}(t)}{(1-\alpha) R_u(t)}.$$

Algorithm 1 Branching Module Algorithm

```
Step 1: Set the initial resource block assignment $\{x'_{u,b,m}\}$
    Discard those $\{u,m\}$ pairs for which $p_{u,m}/c_{u,m} < 0.8\ P/C$
    $\{u',m'\} \leftarrow$ select the $\{u,m\}$ with the largest $p_{u,m}$
        for each RAP among remaining pairs
    $\{x'_{u,b,m}\} \leftarrow$ assign all the resource blocks of the RAP to user
    $u'$ under MCS $m'$
Step 2: Determine whether $P^* > 0.4\ P$ or $P^* \leq 1.6\ P$ holds
    $P' = \sum_{\{u,b,m\}} x'_{u,b,m} p_{u,m}$
            ▷ performance in terms of profit for the $\{x'_{u,b,m}\}$ assignment
    If $P' > 0.8\ P$ then it holds $P^* > 0.4\ P$
    If $P' \leq 0.8\ P$ then it holds $P^* \leq 1.6\ P$
Step 3: Build the resource block assignment solution $\{\tilde{x}_{u,b,m}\}$
    $C' = \sum_{\{u,b,m\}} x'_{u,b,m} c_{u,m}$
    If $C' \leq C$ then return $\{\tilde{x}_{u,b,m}\} = \{x'_{u,b,m}\}$ and stop
    Else ▷ Build a subset of $\{x'_{u,b,m}\}$ denoted by $\{\tilde{x}_{u,b,m}\}$
        $\{\tilde{x}_{u,b,m}\} = \emptyset$
        While $\tilde{C} \doteq \sum_{\{u,b,m\}} \tilde{x}_{u,b,m} c_{u,m} \leq C$ do
            $\{\tilde{x}_{u,b,m}\} \leftarrow$ element remaining in $\{x'_{u,b,m}\}$ with
                the largest $p_{u,m}/c_{u,m}$
        If $\tilde{C} > C/2$ then return $\{\tilde{x}_{u,b,m}\}$ and stop
        Else                                    ▷ Rebuild $\{\tilde{x}_{u,b,m}\}$
            $\{\tilde{x}_{u,b,m}\} = \emptyset$, $\{\tilde{x}_{u,b,m}\} \leftarrow$ element in $\{x'_{u,b,m}\}$
                with the largest $p_{u,m}$
            While $\tilde{C} \leq C$ do
                $\{\tilde{x}_{u,b,m}\} \leftarrow$ element in $\{x'_{u,b,m}\}$ with largest
                $p_{u,m}/c_{u,m}$
            Return $\{\tilde{x}_{u,b,m}\}$ and stop
```

With the above approximation, the optimal solution is given by the $\{x_{u,b,m}\}$ setting that solves the following optimization problem:

$$\underset{\{x_{u,b,m}\}}{\text{maximize}} \sum_{u}\sum_{b}\sum_{m} x_{u,b,m} p_{u,m}$$
$$\text{subject to (2b)} - \text{(2f)},$$

where we define the profit as $p_{u,m} \doteq \log\left(\frac{\alpha r_{u,m}(t)}{(1-\alpha)R_u(t)} + 1\right)$.

To solve the above optimization, the design of CARES consists of three modules: (i) the branching module; (ii) the binary search module, and (iii) the improvement module. CARES works as follows. First, it runs the binary search module to find a candidate solution. In each of the steps of the binary search, the branching module provides a possible solution. Once the binary search returns the candidate solution, the improvement module is executed to try to find a better one. Below, we describe each of the modules of CARES in detail:

1) Branching Module: Let P^* denote the maximum profit, $P^* = \max\left(\sum_{\{u,b,m\}} x_{u,b,m} p_{u,m}\right)$, resulting from finding the maximum value of $\sum_{\{u,b,m\}} x_{u,b,m} p_{u,m}$ under constraints (2b)-(2f). The aim of this module is to determine, for a given P value, if $P^* \leq 1.6P$ or $P^* \geq 0.4\ P$. Moreover, it returns a feasible solution, $\{\tilde{x}_{u,b,m}\}$; hereafter, we let $\tilde{P}$ denote $\sum_{\{u,b,m\}} \tilde{x}_{u,b,m} p_{u,m}$ for the returned subset. The pseudocode is provided in Algorithm 1.

2) Binary Search Module: This module performs a binary search on P until it finds a value that satisfies certain conditions, and returns the solution $\{x_{u,b,m}\}$ corresponding to this value. When the binary search module stops, it returns the solution stored in variable $\hat{x}_{u,b,m}$, whose (aggregate) profit is given by $\hat{P}$. The pseudocode is provided in Algorithm 2.

Algorithm 2 Binary Search Module Algorithm

```
1: Remove the $\{u,m\}$ pairs for which $c_{u,m} > C$
2: $L \leftarrow \max_{u,m} p_{u,m}$, $U \leftarrow B|\mathcal{S}| \max_{u,m} p_{u,m}$, $P \leftarrow U/2$
3: $\hat{P} \leftarrow L$ and $\hat{x}_{u,b,m} \leftarrow$ user with largest $p_{u,m}$ scheduled
                                        ▷ assignment yielding $\hat{P}$
4: While $U/L > 5$ do
5:     Run the branching module to
           determine if $P^* \leq 1.6\ P$ or $P^* \geq 0.4\ P$
           get $\{\tilde{x}_{u,b,m}\}$
6:     If $\tilde{P} > \hat{P}$ then $\{\hat{x}_{u,b,m}\} \leftarrow \{\tilde{x}_{u,b,m}\}$, $\hat{P} \leftarrow \tilde{P}$
7:     If $P^* \leq 1.6\ P$ then $U \leftarrow 1.6P$ and go to 9
8:     If $P^* \geq 0.4\ P$ then $L \leftarrow 0.4P$ and go to 9
9:     $P \leftarrow U/2$
```

3) Improvement Module: The objective of this module is to further improve performance when the solution $\hat{x}_{u,b,m}$ returned by the binary search module does not use all the available resource blocks. The idea is to downgrade the MCS of some selected users (i.e., choosing lower MCS for them) to make room for other users to be scheduled. This module runs in iterations: at each iteration, we obtain a new solution and compare its performance with the previous iteration. We stop when the current iteration does not improve performance over the previous one. The pseudocode is provided in Algorithm 3.

One of the key features of CARES is that it provides a performance bound. The rationale to achieve this bound is as follows. CARES runs the binary search module within an upper limit U and a lower limit L. In each step, we set $P = U/2$ and run the branching module to find a solution $\{x'_{u,b,m}\}$. Based on the outcome of the branching module, we either set $L = 0.4P$ or $U = 1.6P$ and continue the binary search. It can be proven that in all the steps of the search it holds that (i) the profit of the solution provided by the branching algorithm satisfies $P' > L$, and (ii) the upper bound satisfies $U > P^*$. Then, by executing the search until $U/L \leq 5$, we can bound the distance between the profit provided by CARES (P') and the maximum profit (P^*). Note that the choice of the various parameters (such as 0.4 and 1.6) is crucial to guarantee the provided performance bound. This is further analyzed in the following section.

B. Performance Bound

To gain insight into the performance provided by CARES, in the following we provide a theoretical bound for the distance between CARES' performance and the optimal one. Note that, when $\alpha \rightarrow 0$, the approximation of (4) becomes exact and the performance of the optimal solution is equal to the

Algorithm 3 Improvement Module Algorithm

```
Step 1: First iteration
  Take the solution provided by the binary search module
  Add $\{u, m\}$ pairs that fit within the leftover capacity in order of decreasing $p_{u,m}$
  primary users ← users selected by the binary search module
  secondary users ← users added above
  Compute the performance of this solution in terms of aggregate profit
Step 2: Start a new iteration
  Select the user with the largest $c_{u,m}$ and downgrade her MCS
  Remove $\{u, m\}$ pairs with a higher complexity than the old $c_{u,m}$ of the downgraded user
  Keep fixed the MCS of the user downgraded above and of all the other primary users
  Fill the remaining resource blocks following step 3 of the branching module
Step 3: Fill leftover capacity with secondary users
  If some resource blocks remain unassigned then
      Add $\{u, m\}$ pairs that fit within the leftover capacity in order of decreasing $p_{u,m}$
                                  ▷ Update both sets for the next iteration
  primary users ← users selected in step 2
  secondary users ← users selected in step 3
Step 4: Evaluate performance and decide whether to perform a new iteration
  Evaluate performance with the solution provided by step 3
  Compare performance with the one obtained in the previous iteration
  If performance has improved then go to Step 2
  Else stop the algorithm and return the solution from the previous iteration
```

maximum profit P^*. Theorem 3 below shows that under these conditions the proposed algorithm provides bounded performance guarantees; specifically, the (relative) difference between the performance provided by CARES and the optimal one is bounded by a constant factor. The proof of the theorem is provided in Appendix III.

Theorem 3: Let $\{\hat{x}_{u,b,m}\}$ be the solution provided by CARES and let $\hat{P}$ be the profit provided by this solution, i.e., $\hat{P} = \sum_{u \in \mathcal{U}} \sum_{b \in \mathcal{B}_{p(u)}} \sum_{m \in \mathcal{M}_u} \hat{x}_{u,b,m} p_{u,m}$. Then, it holds $\frac{P^* - \hat{P}}{P^*} \leq 4/5$.

C. Complexity Analysis

In the following we analyze the algorithm's complexity, showing that it is upper bounded by $O(|\mathcal{W}| \log(B|\mathcal{S}|) + |\mathcal{W}|^2)$, where $|\mathcal{W}|$ is the number of $\{u, m\}$ pairs in the network satisfying $c_{u,m} \leq C$, $|\mathcal{S}|$ is the number of RAPs and B is the number of resource blocks per RAP. Moreover, the simulations of Section V-F show the actual complexity is substantially lower than this bound. These results confirm that CARES scales well with the size of the network and that computational complexity is sufficiently low to allow its deployment in practical settings, specially taking into account that CARES is expected to run in the cloud, where computational resources are typically not scarce.

The computational complexity bound is computed as follows. For the binary search module, initially we have $L \leq P^* \leq U = B|\mathcal{S}|L$. Every time we visit step 3.1, U is reduced to 0.8 of its previous value. Then, we need no more than $\log_a(B|\mathcal{S}|/5)$ visits to step 3.1 until $U \leq 5L$ (where $a = 10/8$). If at any time the algorithm visits step 3.2, L becomes 0.2 U and the algorithm stops. Thus, we run the branching module no more than $\log_a(B|\mathcal{S}|/5)$ times. Taking into account that the complexity of the branching module is $O(|\mathcal{W}|)$, this yields to an overall complexity of $O(|\mathcal{W}| \log(B|\mathcal{S}|))$, until the binary search module provides a first candidate solution. After the binary search is complete, the improvement module runs at most one iteration for each $\{u, m\}$ pair, since we can downgrade each pair only once. Every iteration incurs a complexity of $O(|\mathcal{W}|)$. Thus, the overall complexity of this module is $O(|\mathcal{W}|^2)$. Adding this value to the one obtained for the binary search module gives the total complexity of the algorithm.

V. Performance Evaluation

We next evaluate via simulation the performance of the CARES algorithm in terms of utility, throughput and capacity savings. In line with similar works in the literature [22], [28], [41]–[44], in our simulator the wireless channel follows the log-distance path loss model with Rayleigh fading [45], with a path loss exponent $\gamma = 3$ and $SNR_0 = 17.6$ dB (which is the SNR threshold for the highest MCS [13]). We set $\alpha = 0.01$ for all experiments (unless otherwise stated) and and employ 95% confidence intervals below 1% (unless explicitly shown). In order to show the gains provided by the CARES algorithm proposed here, we compare its performance against the following benchmarks:

- '*Optimal*', which corresponds to the maximum step algorithm of Section III.
- '*PF w/ MCS*', which first selects users by means of a proportional fair scheduler [17] and then adjusts their MCS following the approach of [10].
- '*Greedy*', which implements the algorithm proposed in [46] to solve the knapsack problem resulting from the approximation given by (4).
- '*Legacy PF*', which implements a legacy proportional fair scheduling [17] to select user transmissions and processes as many frames as possible with the given computational capacity, dropping the frames that exceed the capacity.

A. Utility Performance

We first analyze performance in terms of the objective function (2a), which corresponds to proportional fairness

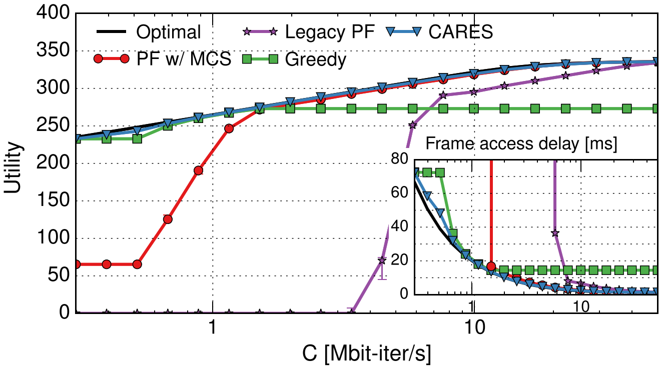

Fig. 2. Utility and access delay performance in a small scale scenario.

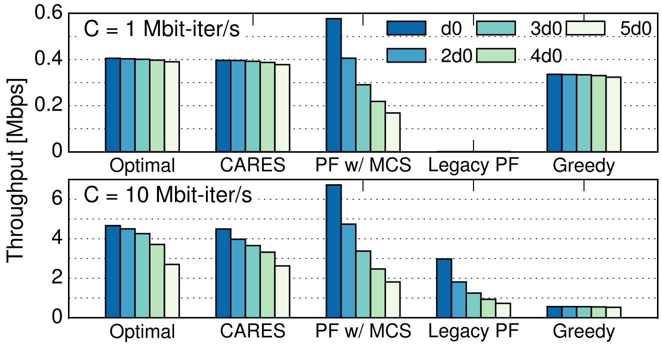

Fig. 3. Throughput distribution for two different C values.

utility [19], comparing the utility provided by CARES against the other approaches. We consider a scenario sufficiently small to allow for the computation of the optimal solution: a network consisting of 5 RAPs, each with 5 users located at a distances $d_0, 2d_0, \ldots, 5d_0$ from the RAP respectively (where d_0 corresponds to the distance that yields an average SNR equal to SNR_0). Each base station has $B = 50$ resource blocks.

Results are shown in Figure 2 as a function of the computational capacity C. They show that CARES closely follows the optimal performance way beyond the theoretically guaranteed bound, confirming that our algorithm is effective in jointly optimizing scheduling and MCS selection. We also observe that CARES substantially outperforms all the other approaches, providing a more graceful degradation as C decreases. The performance obtained with the 'PF w/ MCS' approach is substantially lower for smaller C values, where user selection is heavily constrained by the available capacity (see Sections V-B and V-C for more details). 'Greedy' tends to schedule users with higher $p_{u,m}/c_{u,m}$ ratios, which is a valid strategy when the capacity is low, but does not fully utilize the available capacity when there is plenty of it. Finally, performance with 'Legacy' PF is drastically degraded, as the number of frames that can be decoded rapidly decreases as we reduce capacity, reaching a point where no frames are decoded. As expected, when capacity is sufficient to decode all frames, all approaches (except for 'Greedy') perform optimally, as they all simply apply an optimal PF scheduler with no capacity constraints.

While the main objective of the paper is to provide PF, which mainly concerns throughput performance, delay is also relevant for many applications. In order to provide insight into the delay performance of CARES, the subplot of Figure 2 shows the average frame access delay provided by the different approaches, corresponding to the time elapsed from the instant when a frame reaches the head of the transmission queue until it is transmitted. We observe that CARES substantially outperforms all other approaches and performs close to the optimal benchmark.

B. Throughput Distribution

In order to gain further understanding on the utility gains obtained in the previous section, in Figure 3 we analyze the individual throughput of each user, depending on the user's distance to the base station ($d_0, 2d_0, \ldots, 5d_0$), for two representative capacity values: $C = 1$ Mbit-iter/s and $C = 10$ Mbit-iter/s. We observe from the results that the CARES algorithm provides almost the same throughput distribution as the optimal approach, achieving larger throughputs and/or fairer distributions than the other approaches. It is particularly interesting to observe the behavior of the 'PF w/ MCS' algorithm: by decoupling scheduling from MCS selection, this strategy favors the users with higher SNR when scheduling frames; however, when the MCS of such users is downgraded, this decision turns out to be harmful for the fairness without improving the overall throughput. As anticipated, the 'Greedy' approach performs poorly for $C = 10$ Mbit-iter/s, as it selects users that (although they may be very efficient in relative terms) do not exploit all the available capacity. Finally, it is also interesting to observe that for $C = 1$ Mbit-iter/s the 'Legacy PF' approach is unable to decode any frame, as it only schedules users with very high computational load and does not downgrade their MCS.

C. MCS Downgrades

One of the key ideas behind CARES is that, under shortage of available computational resources, we select MCS schemes that are lower (and computationally less intensive) than the highest possible MCS under current radio conditions. Another approach that follows the same idea is the 'PF w/ MCS', which also downgrades the MCS of scheduled users (until they fit within the available capacity).

Obviously, when applying the above idea, the common goal of both approaches is to select the highest MCS schemes under the available computational capacity. To understand how well do these approaches meet this goal, Figure 4 depicts the average number of downgrades performed by both approaches, i.e., we compare the MCS selected for the transmitted frames against the highest possible MCS for those frames under the current radio conditions. As expected, the average number of downgrades decreases with the computational capacity, as larger capacities allow to fit higher MCS schemes. We further observe from the figure that CARES substantially outperforms 'PF w/ MCS', providing significantly lower downgrade rates. Indeed, as CARES takes computational needs into account when performing scheduling, it can select combina-

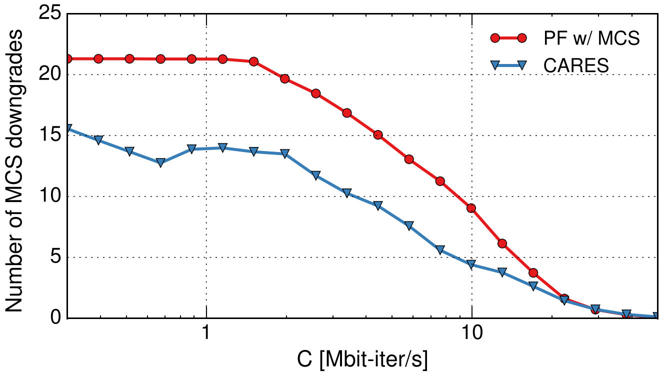

Fig. 4. Average number of downgrades with respect to the highest possible MCS.

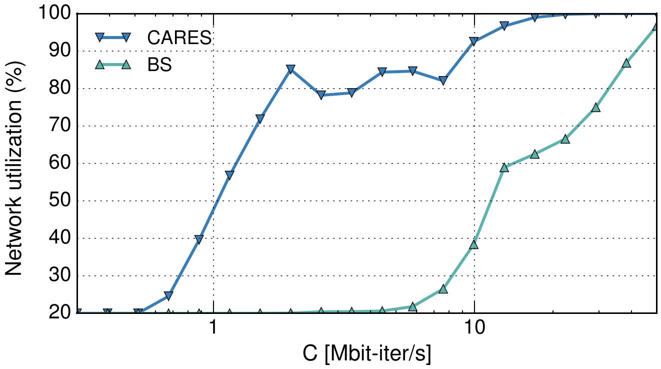

Fig. 5. Network utilization with CARES and with the binary search module solution ('BS').

tions of users that yield not too high aggregated computational load, thus limiting the number of required downgrades. In contrast, scheduling in 'PF w/ MCS' is blind to computational load, and therefore may lead to scheduling combinations that require a larger number of downgrades to fit within C.

D. Network Utilization

One of the design goals of CARES is to maximize network utilization, using as many resource blocks as possible, as this brings advantages in terms of fairness and delay, among other aspects. The improvement module has been specifically designed to achieve this goal. In order to assess its effectiveness, Figure 5 illustrates the fraction of resource blocks utilized by CARES and by the solution provided by the binary search module, respectively. Note that the latter is quite similar to the algorithm of [40], although (as explained in Section IV-A) the algorithm of [40] has been designed for a different problem and hence is not directly applicable to scheduling in a mobile network. Results show the effectiveness of CARES and its advantages over approaches such as [40], which have been devised for more general knapsack problems without taking into account the specific context of our scheduling problem.

E. Impact of the α Parameter

Since our goal in this paper, following the PF criterion, is to optimize the long-term throughputs, we have focused on small α values; as a matter of fact, the optimality proof in Section III relies on this assumption. However, the proposed CARES algorithm can be applied to any α value, including large ones. In order to gain insight into the performance of CARES for different α's, Figure 6 shows the utility provided by the different approaches for a wide range of values. We observe from the results that CARES outperforms the other approaches for all α's. We also observe that, for the 'PF w/ MCS' approach, utility improves as α grows; this is due to the fact that large α values favor the selection of users with lower MCS, and this avoids downgrades.

F. Large Scale Network

In the following we consider a much larger scale scenario, consisting of 50 RAPs serving 50 users each, placed at a random distance between d_0 and $5d_0$ from the RAP. Figure 7 shows the utility performance achieved in this scenario by the various approaches (except for the optimal one). We observe here a similar trend to the one shown by the small scale scenario in Figure 2: the CARES algorithm provides a graceful degradation as capacity decreases, substantially outperforming all other approaches. This confirms the advantages of the proposed approach in realistic (large-scale) scenarios.

In order to understand the computational complexity incurred by CARES in such a large scale scenario, we have measured the number of operations it performs as compared to the upper bound computed in Section IV-C. The results, shown in the subplot of Figure 7, confirm that the actual complexity of the algorithm is very far from the upper bound (in more than two orders of magnitude), which further supports the feasibility of CARES in terms of computational complexity.

We further analyzed a simplified version of the CARES algorithm, called S-CARES, which runs the improvement module algorithm designed in Section IV over the solution provided by the 'Legacy PF' algorithm rather than the one provided by the binary search module. Results show that the performance of this approach is fairly close to CARES while computational complexity is reduced by one order of magnitude, which means that S-CARES can be a viable alternative for environments with heavily constrained computational resources.

G. Savings in Computational Capacity

By making RAN functions robust against computational outages, CARES allows to reduce the amount of computational resources allocated to those functions while providing minimum degradation in performance. In the following, we analyze the capacity savings provided by CARES over the current RAN stack (that is, the 'Legacy PF' approach).

As an intuitive measure to assess performance degradation, we define the relative performance X as follows. Let us consider a reference throughput distribution $\mathbf{r}$, and let $U(\mathbf{r})$ be the corresponding utility. We say that an alternative throughput distribution $\mathbf{r}'$ provides a relative performance X if the utility it provides is equivalent to multiplying by X the throughput of each individual user in the reference distribution,

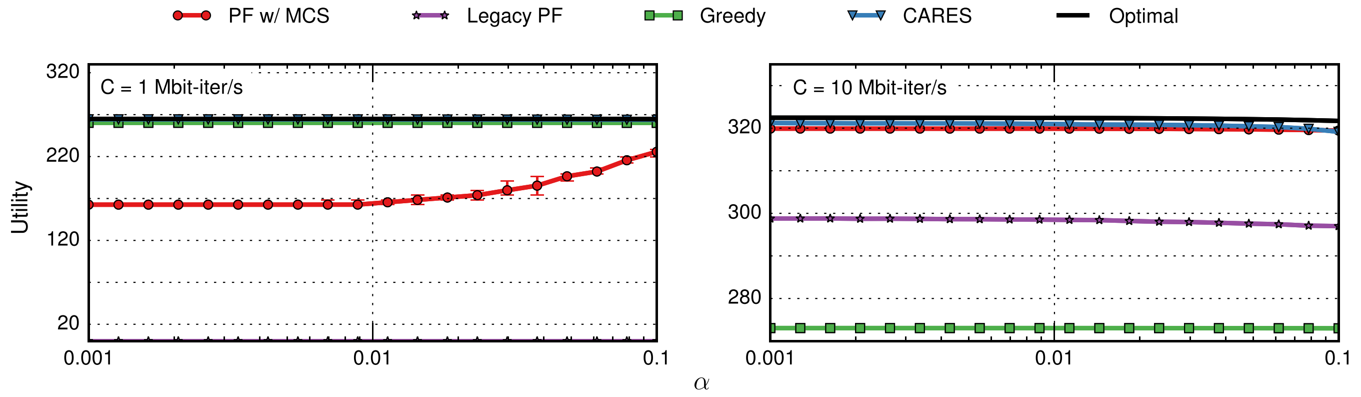

Fig. 6. Utility performance for different α values.

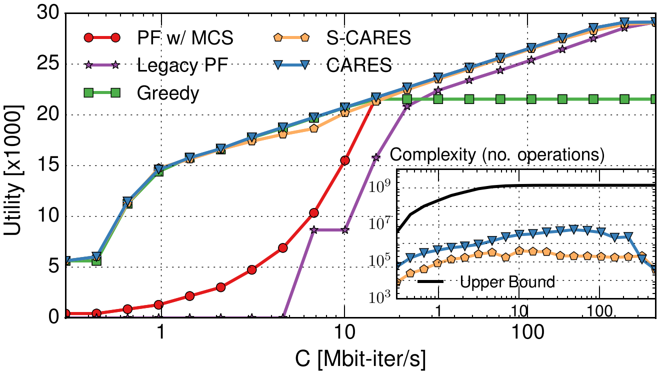

Fig. 7. Utility and complexity in a large scale scenario.

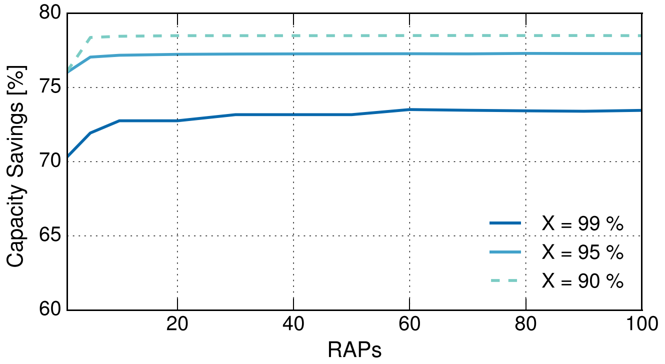

Fig. 8. Computational capacity savings provided by CARES over ‘Legacy PF’.

i.e., $U(\mathbf{r}') = U(\tilde{\mathbf{r}})$, where $\tilde{r}_u = X r_u \ \forall u$ (for instance, a relative performance of $X = 99\%$ is equivalent to multiplying all throughputs by 0.99).

For a given relative performance level X, we obtain the (computational) *capacity savings* achieved by CARES as follows. Let C_{max} be the computational capacity required to ensure that we are always able to decode all frames under the ‘Legacy PF’ approach, and let U_{max} be the utility achieved in this case. Let C_x be the capacity required by the CARES algorithm to provide a relative performance of X with respect to the maximum utility U_{max}. Then, the capacity savings provided by CARES is given by $\frac{C_{max}-C_x}{C_{max}}$, which reflects the amount of capacity that can be saved (in %) when admitting a relative performance degradation of X.

Figure 8 shows the capacity savings achieved by CARES for different relative performance levels, $X = 99\%$, 95% and 90%, as a function of the number of RAPs in the network, under the same conditions as the large scale scenario described in Section V-F. We observe that our approach provides very substantial capacity savings (between 70% and 80%), while paying a very small price in performance. We further observe that, while capacity savings increase when clustering more RAPs in the same pool as a result of the multiplexing gains, they are already very high for a single RAP and almost maximum with only around 5 RAPs. This implies that it is not necessary to cluster a large number of RAPs in a central data-center to achieve high capacity savings; instead, it is sufficient to cluster a small number of RAPs in a location that can be relatively close to the antennas, as would be the case for instance with multi-access edge computing (MEC).

The above results highlight the advantage of employing CARES in a virtualized RAN environment: (i) by making RAN VNFs *computationally aware*, CARES provides very remarkable capacity savings; and (ii) such savings are already achieved with relatively small RAP clusters. Thus, this technology allows an Infrastructure Provider to deploy edge infrastructure (which usually entails high costs) with a very high yield of decoded bytes per deployed CPU cycle ratio.

VI. Conclusions

There is a widely accepted trend towards the virtualization of RAN functions and their execution in cloud-computing platforms. However, the current RAN protocol stack was not designed to be executed in a virtualized environment, and its performance degrades drastically when the required computing resources are not available. To address this issue, in this paper we have re-designed two fundamental functions of the RAN protocol stack: (i) *scheduling*, and (ii) *MCS selection*. By considering computational load when scheduling frame transmissions, we aim at reducing the level of variability of the computational load incurred. Furthermore, by downgrading the selected MCS when needed, we can reduce the impact of computational resource shortage on performance.

One of the key contributions of this paper has been to address the design of the two schemes *jointly*, which leads to substantial performance improvements. When addressing the joint optimization of scheduling and MCS selection, we have relied in analytical tools to obtain the solution that provides optimal performance. Building on the insights gained from this solution, we have designed the CARES algorithm, which incurs limited complexity while providing a performance that is provably close (by a certain factor) to the optimal.

The conducted performance evaluation has confirmed that CARES is a practical and effective approach for virtualizing the RAN functionality. With CARES, we can reduce the amount of computing resource allocated to RAN functions by 70% or more, while retaining almost the entire utility. Furthermore, performance with CARES is gracefully degraded in case of (temporary) resource shortages, such that improper dimensioning of virtual machines pools have a very mild impact in performance. In summary, CARES represents a pioneering step towards a new paradigm of *virtualization-enabled protocol stacks*.

One of the main contribution in this paper has been the proposal of a new paradigm for scheduling algorithms: *computational-aware scheduling*, which takes into account computational resources and jointly optimizes scheduling and MCS selection. While here we have taken a proportional fair scheduler as baseline and have extended it to account for computational resources, the same concept could be applied to other scheduling algorithms (e.g, a scheduler that optimizes delay performance and/or provides minimum rate guarantees), which could be extended following a similar methodology to the one employed in this paper.

Appendix I
Proof of Theorem 1

Theorem 1: If the wireless channels of the RAPs behave according to a Markov chain, then the maximum step algorithm becomes (asymptotically) optimal as $\alpha \to 0$.

Proof: To prove the theorem, we start by showing that as $\alpha \to 0$ the maximum step algorithm defined by our paper is equivalent to the gradient algorithm analyzed in [33]. If we define $U(t) = \sum_{u \in \mathcal{U}} \log R_u$, the gradient is given by

$$\nabla U(t) = \left(\frac{1}{R_0(t)}, \ldots, \frac{1}{R_{|\mathcal{U}|}(t)} \right).$$

The gradient algorithm is the one that maximizes (see [33])

$$G = \nabla U(t)(R_0(t+1) - R_0(t), \ldots, R_{|\mathcal{U}|}(t+1) - R_{|\mathcal{U}|}(t)).$$

Noting that $R_u(t+1) - R_u(t) = \alpha n_{u,m}(t) r_{u,m}(t) - \alpha R_u$, we have

$$G = \sum_{u \in \mathcal{U}} \sum_{m \in \mathcal{M}_u(t)} \frac{\alpha n_{u,m}(t) r_{u,m}(t)}{R_u(t)} - \alpha |\mathcal{U}|.$$

Given that $\alpha|\mathcal{U}|$ is a constant, maximizing G is equivalent to maximizing $\sum_{\{u,m\}} \frac{\alpha n_{u,m}(t) r_{u,m}(t)}{R_u(t)}$. Hence, we conclude that the gradient algorithm is the one that maximizes $\sum_{\{u,m\}} \frac{\alpha n_{u,m}(t) r_{u,m}(t)}{R_u(t)}$.

We now look at the maximum step algorithm and show that it maximizes the same expression as the gradient algorithm. If we take the $\log$ of the function maximized by this algorithm, i.e., (3), and let $\alpha \to 0$, we can see that this algorithm maximizes the following expression

$$\begin{aligned}
&\lim_{\alpha \to 0} \log \left(\prod_{u \in \mathcal{U}} \prod_{m \in \mathcal{M}_u(t)} \left(\frac{\alpha n_{u,m}(t) r_{u,m}(t)}{(1-\alpha) R_u(t)} + 1 \right) \right) \\
&\quad = \lim_{\alpha \to 0} \sum_{u \in \mathcal{U}} \sum_{m \in \mathcal{M}_u(t)} \log \left(\frac{\alpha n_{u,m}(t) r_{u,m}(t)}{(1-\alpha) R_u(t)} + 1 \right) \\
&\quad = \sum_{u \in \mathcal{U}} \sum_{m \in \mathcal{M}_u(t)} \frac{\alpha n_{u,m}(t) r_{u,m}(t)}{(1-\alpha) R_u(t)},
\end{aligned}$$

which (given that the constant $1 - \alpha$ does not change the maximization) coincides with the expression of the gradient algorithm. This confirms that both algorithms are maximizing the same expression and hence are the same algorithm.

We next show that the gradient algorithm is asymptotically optimal. According to [33], the gradient algorithm is asymptotically optimal as long as the following conditions are satisfied: (i) the objective function U is a strictly concave smooth function, (ii) for each element, there is always some decision under which its rate is non-zero, and (iii) the region of long-term rates R_u is convex, bounded and closed. Note that [33] relies on a model that covers our system: it allows that at a given state m (in our case the channel conditions) there are a set $k \in \mathcal{K}$ of scheduling options (in our case the users scheduled in each resource block and the selected MCS) each of which provides a service rate vector $(\mu_1^m(k), \ldots, \mu_{|\mathcal{U}|}^m(k))$ (where $\mu_u^m(k)$ is the rate of user u with the number of resource blocks and MCS corresponding to this scheduling option).

To show that the three conditions of [33] are satisfied by our system, we proceed as follows. Condition (i) holds for our objective function, $U = \sum_{u \in \mathcal{U}} \log R_u$; indeed, this function is precisely one of the functions explicitly considered in [33], referred to as 'Type II Utility Function'. Condition (ii) follows from the fact that when we schedule some user, this user experiences a positive rate. Finally, condition (iii) holds under some natural non-restrictive assumptions that are satisfied by our system model (see [33]).

Since all conditions are satisfied, the result of [33] applies, and the gradient algorithm is (asymptotically) optimal as $\alpha \to 0$. Since, as we have shown above, the maximum step algorithm and the gradient algorithms provide the same solution as $\alpha \to 0$, this means that the maximum step algorithm is asymptotically optimal, which proves the theorem. $\square$

Appendix II
Proof of Theorem 2

Theorem 2: The maximum step algorithm is NP-hard.

Proof: The reduction is via the multiple-choice knapsack (MCKP) problem, which is known to be NP-hard. Recall that the multiple-choice knapsack problem is a variant of the knapsack problem and is stated as follows. We have m

mutually disjoint classes $N_1, \ldots, N_m$ of n_i *items* to pack in a knapsack of *capacity* W. Each item i of class j, $i \in N_j$, has a *profit* p_{ij} and a *weight* w_{ij}, and the problem is to choose at most one item from each class in such a way that (i) the total profit sum is maximized, and (ii) the sum of weights does not exceed W. The MCKP can thus be formulated as:

$$\begin{aligned} \underset{\{x_{ij}\}}{\text{maximize}} \quad & \sum_{j=1}^{m} \sum_{i \in N_j} p_{ij} x_{ij} \\ \text{subject to:} \quad & \sum_{j=1}^{m} \sum_{i \in N_j} w_{ij} x_{ij} \le W, \\ & \sum_{i \in N_j} x_{ij} \le 1 \quad \forall j, \\ & x_{ij} \in \{0,1\} \quad \forall j, i \in N_j. \end{aligned}$$

Let us define an instance of our algorithm where we have: $(i$)one resource block per RAP, (ii) a RAP s for each class N_j, (iii) only one MCS available for each user, (iv) for each item i of class N_j, a user u at RAP s such that $\log\left(\frac{\alpha r_{u,m}(t)}{(1-\alpha)R_u(t)} + 1\right) = p_{ij}$ and $c_{u,m}(t) = w_{ij}$, and (v) the total computational capacity C equal to W.

If now we take a solution to the maximum step algorithm and choose the items that correspond to the users scheduled for transmission, the resulting set of items provides a solution to the MCKP, which proves the theorem. □

Appendix III
Proof of Theorem 3

To prove Theorem 3, we first prove the two Lemmas below.

Lemma 1: Let P' be the value computed by the branching module for a given P. If $P' > 0.8\ P$ then $P^* > 0.4\ P$, and if $P' \le 0.8\ P$ then $P^* \le 1.6\ P$

Proof: Suppose $P' > 0.8\ P$. Consider the following two cases: (i) $\sum_{\{u,b,m\}} x'_{u,b,m} c_{u,m} \le C$, and (ii) $\sum_{\{u,b,m\}} x'_{u,b,m} c_{u,m} > C$.

In case (i), $\{x'_{u,b,m}\}$ provides a feasible solution to the problem satisfying $\sum_{\{u,b,m\}} x'_{u,b,m} p_{u,m} > 0.8\ P$ and hence $P^* > 0.8\ P > 0.4\ P$.

In case (ii), we choose a subset $\mathcal{B}'$ of resource blocks, and an additional resource block b' (assigned to user u' under MCS m') such that $\sum_{\{b \in \mathcal{B}', u, m\}} x'_{u,b,m} c_{u,m} < C$ and $\sum_{\{b \in \mathcal{B}', u, m\}} x'_{u,b,m} c_{u,m} + c_{u',m'} > C$. Given that $p_{u,m}/c_{u,m} \ge 0.8\ P/C$ for $\{x'_{u,b,m}\}$, we have

$$\begin{aligned} &\sum_{\{b \in \mathcal{B}', u, m\}} x'_{u,b,m} p_{u,m} + p_{u',m'} \\ &\qquad \ge \frac{0.8P}{C} \left(\sum_{\{b \in \mathcal{B}', u, m\}} x'_{u,b,m} c_{u,m} + c_{u',m'} \right) > 0.8P. \end{aligned}$$

Further, $P^* \ge \sum_{\{b \in \mathcal{B}', u, m\}} x'_{u,b,m} p_{u,m}$ and $P^* \ge p_{u',m'}$, which implies that

$$2P^* > \sum_{\{b \in \mathcal{B}', u, m\}} x'_{u,b,m} p_{u,m} + p_{u',m'} > 0.8P$$

and therefore $P^* > 0.4P$.

Now suppose that $P' \le 0.8\ P$. Let $x^*_{u,b,m}$ be the optimal assignment that yields P^*. Consider the optimal assignment, and denote by $\mathcal{B}_1$ (respectively $\mathcal{B}_2$) the set of resource blocks within this assignment for which $p_{u,m}/c_{u,m} < 0.8P/C$ (respectively $p_{u,m}/c_{u,m} \ge 0.8P/C$) holds. Then,

$$\begin{aligned} P^* &= \sum_{\{u,b,m\}} x^*_{u,b,m} p_{u,m} \\ &= \sum_{\{b \in \mathcal{B}_1, u, m\}} x^*_{u,b,m} p_{u,m} + \sum_{\{b \in \mathcal{B}_2, u, m\}} x^*_{u,b,m} p_{u,m}. \end{aligned}$$

But

$$\sum_{\{b \in \mathcal{B}_1, u, m\}} x^*_{u,b,m} p_{u,m} < \frac{0.8P}{C} \sum_{\{b \in \mathcal{B}_1, u, m\}} x^*_{u,b,m} c_{u,m} \le 0.8P.$$

Further, from our initial assumption and the choice at step 1 of the branching module, we have

$$\sum_{\{b \in \mathcal{B}_2, u, m\}} x^*_{u,b,m} p_{u,m} \le \sum_{\{u,b,m\}} x'_{u,b,m} p_{u,m} \le 0.8P.$$

Therefore $P^* < 1.6P$. □

Lemma 2: When $P' > 0.8\ P$, the setting returned by the branching module satisfies $\tilde{P} > 0.4\ P$.

Proof: If $\sum_{\{u,b,m\}} x'_{u,b,m} c_{u,m} \le C$, we can see from the proof of Lemma 1 that $\tilde{P} = P' > 0.8\ P > 0.4\ P$ holds for the items selected by the branching algorithm. Otherwise, we consider two cases: (i) the sum of the $c_{u,m}$'s of the items selected by the branching module is larger than $C/2$, and (ii) the sum of the $c_{u,m}$'s of these items is smaller than or equal to $C/2$.

For case (i) we have $\sum_{\{u,b,m\}} \tilde{x}_{u,b,m} c_{u,m} > C/2$, and thus it holds

$$\begin{aligned} \tilde{P} &= \sum_{\{u,b,m\}} \tilde{x}_{u,b,m} p_{u,m} \ge \frac{0.8P}{C} \sum_{\{u,b,m\}} \tilde{x}_{u,b,m} c_{u,m} \\ &> \frac{0.8P}{C} \frac{C}{2} = 0.4P. \end{aligned}$$

For case (ii), we proceed as follows. If we take any item $\{u', m'\}$ that belongs to $\{x'_{u,b,m}\}$ but not to $\{\tilde{x}_{u,b,m}\}$, necessarily the sum of the $c_{u,m}$ of this item plus all the items in $\{\tilde{x}_{u,b,m}\}$ exceeds C, as otherwise $\{u', m'\}$ would be part of $\{\tilde{x}_{u,b,m}\}$. Applying the same reasoning as in Lemma 1, we have

$$\sum_{\{u,b,m\}} \tilde{x}_{u,b,m} p_{u,m} + p_{u',m'} > 0.8P.$$

Furthermore, since $\{\tilde{x}_{u,b,m}\}$ comprises the item with the largest $p_{u,m}$, we also have

$$\sum_{\{u,b,m\}} \tilde{x}_{u,b,m} p_{u,m} \ge p_{u',m'}.$$

The above two equations yield $\tilde{P} = \sum_{\{u,b,m\}} \tilde{x}_{u,b,m} p_{u,m} > 0.4P$. □

Theorem 3: Let $\{\hat{x}_{u,b,m}\}$ be the solution provided by CARES and let $\hat{P}$ be the profit provided by this solution, i.e., $\hat{P} = \sum_{u \in \mathcal{U}} \sum_{b \in \mathcal{B}_{p(u)}} \sum_{m \in \mathcal{M}_u} \hat{x}_{u,b,m} p_{u,m}$. Then, it holds $\frac{P^* - \hat{P}}{P^*} \le 4/5$.

Proof: Based on Lemmas 1 and 2, we proceed as follows to prove the theorem. The proof focuses only on the binary search module and does not consider the improvement module. Indeed, as the improvement module just increases the performance provided by the previous steps, any bound obtained without taking into account this module will hold for the complete algorithm.

It can be seen that $\hat{P} \geq L$; indeed, (i) initially we have $\tilde{P} \geq L$; (ii) when we visit step 3.2 of the binary search module, L increases to $L = 0.4\ P$ but from Lemma 2 we have that the resulting $\tilde{P}$ satisfies $\tilde{P} > 0.4\ P$ and hence $\tilde{P} \geq L$ holds; and (iii) when we visit step 3.1, we have that $\tilde{P}$ may increase while L remains constant. Hence, $\tilde{P} \geq L$ holds in any intermediate steps, and will also hold in the last step, i.e., $\hat{P} \geq L$. Furthermore, from Lemma 1 it holds that $U \geq P^*$; indeed, initially $U \geq P^*$, and we only decrease U to $1.6P$ when $P^* \leq 1.6P$.

The algorithm is guaranteed to stop because the length of the interval $[L, U]$ decreases by at least 12.5% every time the algorithm runs steps 3 and 4. When the algorithm stops, we have $U/L \leq 5$ from the stopping conditions. Since $U \geq P^*$, this implies that $P^*/L \leq 5$ and hence $L \geq P^*/5$. Furthermore, since $\hat{P} \geq L$ this yields $\hat{P} \geq P^*/5$, which proves the theorem. □

References

[1] P. Rost *et al.*, "Mobile network architecture evolution toward 5G," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 84–91, May 2016.

[2] *Study on New Radio Access Technology: Radio Access Architecture and Interfaces*, document TR 38.801, Aug. 2016.

[3] A. Checko *et al.*, "Cloud RAN for mobile networks—A technology overview," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 405–426, 4th Quart., 2015.

[4] "C-RAN: The road towards green RAN," China Mobile Res. Inst., Beijing, China, White Paper 2.5, Dec. 2013.

[5] P. Rost *et al.*, "Cloud technologies for flexible 5G radio access networks," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 68–76, May 2014.

[6] D. Wubben *et al.*, "Benefits and impact of cloud computing on 5G signal processing: Flexible centralization through cloud-RAN," *IEEE Signal Process. Mag.*, vol. 31, no. 6, pp. 35–44, Nov. 2014.

[7] M. Madhavan, P. Gupta, and M. Chetlur, "Quantifying multiplexing gains in a wireless network cloud," in *Proc. IEEE ICC*, Jun. 2012, pp. 3212–3216.

[8] T. L. Nguyen and A. Lebre, "Virtual machine boot time model," in *Proc. PDP*, Mar. 2017, pp. 430–437.

[9] *Policy and Charging Control Architecture*, document TS 23.203, 2017.

[10] P. Rost, A. Maeder, M. C. Valenti, and S. Talarico, "Computationally aware sum-rate optimal scheduling for centralized radio access networks," in *Proc. IEEE GLOBECOM*, Dec. 2015, pp. 1–6.

[11] M. C. Valenti, S. Talarico, and P. Rost, "The role of computational outage in dense cloud-based centralized radio access networks," in *Proc. Global Commun. Conf. (GLOBECOM)*, Dec. 2014, pp. 1466–1472.

[12] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "OpenAirInterface: A flexible platform for 5G research," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 33–38, 2014.

[13] P. Rost, S. Talarico, and M. C. Valenti, "The complexity–rate tradeoff of centralized radio access networks," *IEEE Trans. Wireless Commun.*, vol. 14, no. 11, pp. 6164–6176, Nov. 2015.

[14] Y. Li, M. Sheng, X. Wang, Y. Zhang, and J. Wen, "Max–min energy-efficient power allocation in interference-limited wireless networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 9, pp. 4321–4326, Sep. 2015.

[15] Z. Li, S. Guo, D. Zeng, A. Barnawi, and I. Stojmenovic, "Joint resource allocation for max-min throughput in multicell networks," *IEEE Trans. Veh. Technol.*, vol. 63, no. 9, pp. 4546–4559, Nov. 2014.

[16] M. Kalil, A. Shami, and A. Al-Dweik, "QoS-aware power-efficient scheduler for LTE uplink," *IEEE Trans. Mobile Comput.*, vol. 14, no. 8, pp. 1672–1685, Aug. 2015.

[17] M. Andrews, "A survey of scheduling theory in wireless data networks," *Wireless Communications* (The IMA Volumes in Mathematics and Its Applications), vol. 143. Atlanta, GA, USA: Springer, 2007, pp. 1–17.

[18] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushyana, and S. Viterbi, "CDMA/HDR: A bandwidth efficient high speed wireless data service for nomadic users," *IEEE Commun. Mag.*, vol. 38, no. 7, pp. 70–77, Jul. 2000.

[19] T. Bu, L. Li, and R. Ramjee, "Generalized proportional fair scheduling in third generation wireless data networks," in *Proc. IEEE INFOCOM*, Apr. 2006, pp. 1–12.

[20] S. Borst, N. Hegde, and A. Proutiere, "Mobility-driven scheduling in wireless networks," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 1260–1268.

[21] H. J. Kushner and P. A. Whiting, "Convergence of proportional-fair sharing algorithms under general conditions," *IEEE Trans. Wireless Commun.*, vol. 3, no. 4, pp. 1250–1259, Jul. 2004.

[22] P. Viswanath, D. N. C. Tse, and R. Laroia, "Opportunistic beamforming using dumb antennas," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1277–1294, Jun. 2002.

[23] R. Kwan, C. Leung, and J. Zhang, "Proportional fair multiuser scheduling in LTE," *IEEE Signal Process. Lett.*, vol. 16, no. 6, pp. 461–464, Jun. 2009.

[24] J. Fan, Q. Yin, G. Y. Li, B. Peng, and X. Zhu, "MCS selection for throughput improvement in downlink LTE systems," in *Proc. IEEE ICCCN*, Aug. 2011, pp. 1–5.

[25] R. Kwan, C. Leung, and J. Zhang, "Resource allocation in an LTE cellular communication system," in *Proc. IEEE ICC*, Jun. 2009, pp. 1–5.

[26] J. Bartelt, P. Rost, D. Wubben, J. Lessmann, B. Melis, and G. Fettweis, "Fronthaul and backhaul requirements of flexibly centralized radio access networks," *IEEE Wireless Commun.*, vol. 22, no. 5, pp. 105–111, Oct. 2015.

[27] S. Bhaumik *et al.*, "CloudIQ: A framework for processing base stations in a data center," in *Proc. ACM MobiCom*, Aug. 2012, pp. 125–136.

[28] H. A. M. Ramli, R. Basukala, K. Sandrasegaran, and R. Patachaianand, "Performance of well known packet scheduling algorithms in the downlink 3GPP LTE system," in *Proc. IEEE MICC*, Dec. 2009, pp. 815–820.

[29] S. B. Lee, I. Pefkianakis, A. Meyerson, and S. Xu, "Proportional fair frequency-domain packet scheduling for 3GPP LTE uplink," in *Proc. INFOCOM*, Apr. 2009, pp. 2611–2615.

[30] H. Safa and K. Tohme, "LTE uplink scheduling algorithms: Performance and challenges," in *Proc. ICT*, Apr. 2012, pp. 1–6.

[31] A. S. Lioumpas and A. Alexiou, "Uplink scheduling for Machine-to-Machine communications in LTE-based cellular systems," in *Proc. IEEE GLOBECOM Workshops*, Dec. 2011, pp. 353–357.

[32] F. Capozzi, G. Piro, L. A. Grieco, G. Boggia, and P. Camarda, "Downlink packet scheduling in LTE cellular networks: Key design issues and a survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 678–700, May 2013.

[33] A. L. Stolyar, "On the asymptotic optimality of the gradient scheduling algorithm for multiuser throughput allocation," *Oper. Res.*, vol. 53, no. 1, pp. 12–25, Jan. 2005.

[34] J.-T. Tsai, "State-dependent proportional fair scheduling algorithms for wireless forward link data services," in *Proc. IEEE INFOCOM*, Apr. 2008, pp. 2414–2422.

[35] A. Konrad, B. Y. Zhao, A. D. Joseph, and R. Ludwig, "A Markov-based channel model algorithm for wireless networks," *Wireless Netw.*, vol. 9, no. 3, pp. 189–199, 2003.

[36] Q. Zhang and S. A. Kassam, "Finite-state Markov model for Rayleigh fading channels," *IEEE Trans. Commun.*, vol. 47, no. 11, pp. 1688–1692, Nov. 1999.

[37] H. S. Wang and N. Moayeri, "Finite-state Markov channel—A useful model for radio communication channels," *IEEE Trans. Veh. Technol.*, vol. 44, no. 1, pp. 163–171, Feb. 1995.

[38] M. Zorzi, R. R. Rao, and L. B. Milstein, "On the accuracy of a first-order Markov model for data transmission on fading channels," in *Proc. 4th IEEE Int. Conf. Universal Pers. Commun.*, Nov. 1995, pp. 211–215.

[39] J. McDougall and S. Miller, "Sensitivity of wireless network simulations to a two-state Markov model channel approximation," in *Proc. IEEE GLOBECOM*, Dec. 2003, pp. 697–701.

[40] G. Gens and E. Levner, "An approximate binary search algorithm for the multiple-choice knapsack problem," *Inf. Process. Lett.*, vol. 67, no. 5, pp. 261–265, Sep. 1998.

[41] E. Yaacoub and Z. Dawy, "A game theoretical formulation for proportional fairness in LTE uplink scheduling," in *Proc. IEEE WCNC*, Apr. 2009, pp. 1–5.

[42] *Technical Specification Group Radio Access Network; Deployment Aspects*, document TS 25.943, Mar. 2017.

[43] S. Schwarz, C. Mehlführer, and M. Rupp, "Low complexity approximate maximum throughput scheduling for LTE," in *Proc. Conf. Rec. 44th Asilomar Conf. Signals, Syst. Comput. (ASILOMAR)*, Nov. 2010, pp. 1563–1569.

[44] E. Liu and K. K. Leung, "Proportional fair scheduling: Analytical insight under Rayleigh fading environment," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Mar. 2008, pp. 1883–1888.

[45] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

[46] E. K. Burke and G. Kendall, *Search Methodologies*. Atlanta, GA, USA: Springer, 2005.

Dario Bega received the B.Sc. and M.Sc. degrees in telecommunication engineering from the University of Pisa, Italy, in 2010 and 2013, respectively. He is currently pursuing the Ph.D. degree in telematic engineering with the University Carlos III of Madrid. He is also a Research Assistant with the IMDEA Networks Institute, Madrid. His research work focuses on wireless networks, multi-tenancy, and machine learning approach for 5G.

Albert Banchs (M'04–SM'12) received the M.Sc. and Ph.D. degrees from the Polytechnic University of Catalonia (UPC-BarcelonaTech) in 1997 and 2002, respectively. He is currently a Full Professor with the University Carlos III of Madrid (UC3M), and has a double affiliation as the Deputy Director of the IMDEA Networks Institute. Before joining UC3M, he was at ICSI Berkeley in 1997, at Telefonica I+D in 1998, and at NEC Europe Ltd., from 1998 to 2003. His research interests include the performance evaluation and algorithm design in wireless and wired networks. He is an Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and the IEEE/ACM TRANSACTIONS ON NETWORKING.

Marco Gramaglia received the M.Sc. and Ph.D. degrees in telematics engineering from the University Carlos III of Madrid (UC3M) in 2009 and 2012, respectively. He held post-doctoral research positions at the Istituto Superiore Mario Boella, Italy, the Institute of Electronics, Computer, and Telecommunications Engineering (IEIIT), National Research Council, Italy, (CNR, Turin, Italy,) CNR-IEIIT, Italy, and IMDEA Networks, Spain. He is currently a Post-Doctoral Researcher at UC3M. He was involved in several EU projects and has authored over 30 scientific publications in international books, conferences and journals.

Xavier Costa-Pérez (M'01) received the M.Sc. and Ph.D. degrees in telecommunications from the Polytechnic University of Catalonia (UPC-BarcelonaTech). He is currently the Head of 5G Networks R&D at NEC Laboratories Europe, where he manages several projects focused on 5G mobile core, backhaul/fronthaul, and access networks. His team contributes to NEC projects for products roadmap evolution to European Commission R&D collaborative projects and to open-source projects and related standardization bodies, and has received several R&D awards for successful technology transfers. He has served on the program committees of several conferences and holds multiple patents. He was a recipient of the National Award for his Ph.D. thesis.

Peter Rost (S'06–M'10–SM'15) received the M.Sc. degree from the University of Stuttgart, Stuttgart, Germany, in 2005, and the Ph.D. degree from Technische Universität Dresden, Dresden, Germany, in 2009. He has been with the Fraunhofer Institute for Beam and Material Technologies, Dresden; IBM Deutschland Entwicklung GmbH, Böblingen, Germany; and NEC Laboratories Europe, Heidelberg, Germany. Since 2015, he has been a member of the e2e Architecture Research Group at Nokia Bell Labs, Munich, Germany, where he is TM of German BMBF Project TACNET 4.0, and leads the 5G Industrial IOT Testbed at Hamburg Seaport, and works on business unit projects on 5G architecture. He currently serves as a member of the VDE ITG Expert Committee Information and System Theory and is an Executive Editor for the IEEE TRANSACTIONS OF WIRELESS COMMUNICATIONS.