Dynamic slicing of RAN resources for heterogeneous coexisting 5G services

Sihem Bakri, Pantelis A. Frangoudis, and Adlen Ksentini
EURECOM, Sophia Antipolis, France
Email: {name.surname}@eurecom.fr

***Abstract*—Network slicing is one of the key components allowing to support the envisioned 5G services, which are organized in three different classes: Enhanced Mobile Broadband (eMBB), massive Machine Type Communication (mMTC), and Ultra-Reliable and Low-Latency Communication (URLLC). Network Slicing relies on the concept of Network Softwarization (Software Defined Networking - SDN and Network Functions Virtualization - NFV) to share a common infrastructure and build virtual instances (slices) of the network tailored to the needs of different 5G services. Although it is straightforward to slice and isolate computing and network resources for Core Network (CN) elements, isolating and slicing Radio Access Network (RAN) resources is still challenging. In this paper, we leverage a two-level MAC scheduling architecture and provide a resource sharing algorithm to compute and dynamically adjust the necessary radio resources to be used by each deployed network slice, covering eMBB and URLLC slices. Simulation results clearly indicate the ability of our solution to slice the RAN resources and satisfy the heterogeneous requirements of both types of network slices.**

***Index Terms*—5G, Network slicing, Scheduling, Radio resources sharing.**

I. Introduction

The network slicing paradigm has been adopted in the design of current 5G systems as a key technological component. It aims at sharing the same physical infrastructure (Mobile Network infrastructure, RAN and Core Network), by creating virtual instances of the network tailored to application needs. To build these virtual instances on top of the underlying physical substrate, Network Slicing uses Network Softwarization techniques, i.e., Software Defined Networking (SDN), Network Functions Virtualization (NFV) and Cloud Computing. Network slicing requires sophisticated mechanisms to share and isolate resources among the coexisting slices. While the core and transport network resources can be more easily dimensioned and isolated for each slice, RAN resource sharing across slices remains a challenge hard to tackle. In [1] and [2], the concept of a *two-level scheduler* was introduced, which aims to share physical radio resources (i.e., physical Resource Blocks - $pRBs$) among slices by abstracting pRBs and using two scheduler levels: The first level is slice-specific, allowing each slice to use its own internal scheduler, and schedules each User Equipment (UE) with *virtual* Resource Blocks (vRBs). The second level, on the other hand, considers the slice-specific (virtual) resource assignment and maps it to actual pRBs. However, since the number of pRBs (N_{pRB}) is limited, the second-level scheduler controls the number of pRBs assigned to each slice according to the recommendation of a *Slice Orchestrator (SO)*. The latter indicates the maximum number of pRBs to dedicate to each slice, after executing an intra-slice physical resource sharing algorithm.

However, in the above works it is not detailed how these values (N_{pRB}) are derived for each slice, knowing that each slice type has its own characteristics and requirements. For instance, eMBB (enhanced Mobile Broadband) requests high bandwidth, while URLLC (Ultra-Reliable and Low-Latency Communication) aims to minimize latency and maximize reliability. Rather, the ratio of radio resources to allocate to each slice is considered static and is decided in a manner agnostic to the actual application requirements of each slice in terms of latency and/or throughput. This work fills this gap by completing the work of [1] and [2] with a dynamic RAN resource slicing mechanism to derive the value of N_{pRB} to dedicate to each running slice, according to its specific requirements and the varying conditions of the radio environment. The proposed mechanism runs at the SO level, and relies on monitoring information obtained from the RAN.

Our contributions are the following: In § III, using tools from queuing theory, we propose an algorithm that derives the *service rate* necessary to sustain the latency requirements of a URLLC slice, which it translates to an initial pRB allocation. This mechanism relies solely on information obtained from a *slice template* provided by the slice owner, and is agnostic to the channel conditions of each user. Similarly, our mechanism provides an initial pRB allocation for eMBB slices aiming to satisfy their throughput requirements. Then, in § IV, we design a RAN-aware dynamic slicing algorithm that exploits per-user RAN-level information to more accurately translate the derived service rate to an appropriate pRB assignment. Simulation results on the performance of the proposed algorithms follow in § V.

II. Related work

Several works in the literature address the allocation of resources to Network Slices. In [4], the authors discuss the dynamic allocation of RAN resources to different tenants (e.g., virtual mobile network operators and service providers). They propose a weighed proportionally fair allocation mechanism, which aims to ensure the desirable fairness and protection among the network slices of the different tenants and their associated users.

The authors of [5] design optimization algorithms for common scheduling between eMBB and URLLC slice traffic, considering the dual objectives of maximizing utility for eMBB traffic while satisfying instantaneous URLLC requests. This is achieved by dynamically multiplexing the URLLC traffic through puncturing/superposition of the eMBB Traffic. The results show that this joint problem has structural properties

that enable clean decomposition, and corresponding algorithms with theoretical guarantees.

In [6] the authors analyze dynamic resource sharing in network slicing when tenants (such as mobile operators and/or services) support inelastic users with minimum rate requirements. They propose a network slicing framework combining (i) admission control, (ii) resource allocation, and (iii) user dropping, which they study using tools from game theory.

The authors of [7] present algorithms that study the problem of resource allocation in the context of a slicing-ready 5G network. These algorithms are composed by: i) traffic analysis and prediction per network slice using the Holt-Winters forecasting procedure to analyze and predict future traffic requests associated with a particular network slice, ii) admission control decisions for network slice requests using a heuristic algorithm, and iii) adaptive correction of the forecasting solution based on the measured deviations, using a proposed network slice scheduler.

The authors of [8] focus on the computational outages that can occur between RAN functions, aiming to improve the performance of scheduling and modulation and coding scheme (MCS) selection functions. The problem, which was shown to be NP-hard, was formulated as a joint optimization one and some algorithms to solve it were proposed.

Finally, [9] adopts revenue management models, which have been introduced in other contexts (airlines, hotels, etc.), in order to propose a resource allocation model. The authors propose the concept of slice overbooking to maximize mobile operators' revenues, by introducing a hierarchical control plane to manage the orchestration of slices.

To summarize, despite the fact that all the methods already proposed have shown relatively good results in the challenge of dynamic resource allocation, all these methods propose algorithms for dynamic resource sharing individually or in combination, which are based on several constraints on users, operators, etc., which require several information from them, and which can not always be feasible, optimal and/or accurate. On the other hand, our contribution proposes a simpler algorithm which is based only on the estimation of the quality of the channel, as well as it allows to estimate the number of resources to allocate to each slice, which adds more precision to the system.

III. Architecture and assumptions

In this work, we envision the same network architecture model adopted in [1] and [2]. We assume a 5G network which includes a SO and a set of eNodeBs deployed covering an area. The role of the SO is to deploy and manage the life cycle of network slices in the mobile network (RAN and Core Network). We assume that a SO is responsible for a region covered by a certain number of eNodeBs. The SO communicates with the eNodeBs using a southbound protocol, such as FlexRAN [3], that allows to interact and manage remotely the eNodeBs. The eNB management process consists in getting status information on the RAN and appropriately configuring eNodeBs, e.g., by setting the number of pRBs to dedicate to each slice. We assume that a set of UEs are served by/associated with a network slice, spanning a set of eNodeBs (i.e., different physical locations). The SO receives from a tenant (owner) a request to instantiate a slice in the form of a slice template, which indicates the slice type (e.g., eMBB, uRLLC, mMTC), its duration, the list of involved UEs, the (application) data rate (denoted by λ) of the service used in this slice, and application requirements such as the maximum tolerated latency. According to this information, the SO derives the appropriate number of $pRBs$ that fits the needs of the slice, which will be communicated later to the involved eNodeBs via the southbound protocol.

In this work, we consider that a network slice is either eMBB or URLLC; hence, we propose two corresponding mechanisms to estimate the N_{pRB} needed by each slice. Note that although the 5G system considers three types of slices, in this paper we considered only two of them; resource allocation for eMBB and mMTC may follow the same mechanisms and algorithms. The main difference lies between eMBB and uRLLC, as the first one seeks high data rate, while the second requires low latency. The proposed algorithm first derives an initial estimation of the number of pRBs necessary using the information obtained from the slice template. Then, a dynamic algorithm is used to tune N_{pRB} periodically according to the feedback obtained from the eNBs via the southbound protocol. In this section, we will detail the first step for each network slice type considered (eMBB and URLLC).

A. eMBB slice

An eMBB slice requires high data rate, which will represent the main objective when estimating N_{pRB}. In the first step, we start by estimating the maximum number of required $pRBs$ for each eNodeB i ($N_{pRBmax}(i)$), using the information provided by the slice owner, i.e., the data rate per user required by the application running on top of the slice ($d_{App/user}$), and the number of users ($N_{users}(i)$) of the network slice connected to eNodeB i, which can be retrieved from the eNodeB via the southbound control protocol. The main constraint to satisfy is that the number of pRBs $N_{pRBmax}(i)$ to dedicate periodically for an eMBB slice at each eNodeB should be (greater than or) equal to the aggregate data rate needed by the slice application for all users connected to it; this is captured in (1).

$$N_{pRBmax}(i) * d_{pRB} = N_{users}(i) * d_{App/user}. \qquad (1)$$

Indeed, the equation indicates that the $N_{pRBmax}(i)$ allowed to a slice on a given eNodeB i should cover the needed slice's applications (i.e., the number of active users $N_{users}(i)$ multiplied by the data rate required by the application). Here we consider that $d_{App/user}$ is the same for all users. We further assume that d_{pRB} is the maximum data rate provided by one pRB, and that it is the same for all users. In this first step, we consider that this rate is the maximum achievable by the radio system for ideal channel conditions, i.e., the maximum possible Channel Quality Indicator (CQI) value of 15, and the corresponding MCS and transport block size as specified in the standard [10].

Once each $N_{pRBmax}(i)$ is computed using (1), it is communicated via the southbound protocol to the corresponding eNodeBs.

B. URLLC slice

Knowing that a URLLC slice includes all services requiring ultra-low latency, the aim when deriving N_{pRBmax} is to keep latency below a maximum threshold (Lat_{max}) indicated in the slice template provided by the slice owner. To do that, we need to derive a model that estimates the latency experienced by URLLC packets at the eNodeB queue.

Given that each slice has its own downlink queue at the eNodeB [1], all packets belonging to the slice share the same queue. Therefore, to estimate the latency of the packets, we propose to model the slice queue at the eNB as an M/M/1/K one. The traffic arrival rate follows a Poisson distribution with intensity λ, the service rate μ is exponential and the queue has a size of K. Here, the value of λ corresponds to the traffic rate of the application running on top of the slice, while the service rate μ depends on the scheduling process at the MAC layer. To derive λ and μ we can use the following formulas:

$$\mu = \frac{N_{pRB} * d_{pRB}}{avg\_packet\_size} \tag{2}$$

$$\lambda = \frac{N_{users} * d_{App/user}}{avg\_packet\_size}, \tag{3}$$

with avg_packet_size denoting the average packet size of the URLLC application, and $d_{App/_user}$ having the same value for all slice users. To estimate the latency of URLLC packets, we apply Little's law. The latter assumes that whatever the distribution of the arrival rate, the average time a user spends in a queue depends on the number of active users N_{users} and the traffic intensity (i.e., λ). As the number of users corresponds in our case to the number of packets (N_{packet}) of the URLLC service waiting in the queue, Little's law is used as follows to derive the time a packet spends in the queue:

$$T_w = \frac{N_{packet}}{\lambda} \tag{4}$$

As we assumed that the URLLC queue is modeled as M/M/1/K, N_{packet} can be derived as follows:

$$N_{packet} = \frac{1-\rho}{1-\rho^{K+1}} \sum_{k=0}^{K} k\rho^k \tag{5}$$

where $\rho = \frac{\lambda}{\mu}$. Since μ corresponds to the service rate of the URLLC queue, and depends on the number of resources dedicated to the URLLC slice, it can be derived using (2). By assuming that Lat_{max} is the maximum tolerated latency by a URLLC slice, T_w should be less than or equal to this value:

$$T_w \leq Lat_{max}. \tag{6}$$

We substitute T_w by its value given by (4), obtaining the following expression:

$$\frac{N_{packet}}{\lambda} = \frac{\frac{1-\frac{\lambda}{\mu}}{1-(\frac{\lambda}{\mu})^{K+1}} \sum_{k=0}^{K} k(\frac{\lambda}{\mu})^k}{\lambda} \leq Lat_{max} \tag{7}$$

Therefore, we need to find a value of μ, noted μ_{opt}, that ensures at least a latency equal to Lat_{max} for URLLC. According to (2), we can extract the number of pRBs (noted N_{pRBopt}) to dedicate to a URLLC slice as follows:

$$N_{pRBopt} = \frac{\mu_{opt} * avg\_packet\_size}{d_{pRB}} \tag{8}$$

At this step, we go by the assumption that the value of d_{pRB} is the same for all UEs, as in the case of eMBB, and that avg_packet_size is constant, and aim to solve (7) for μ. We denote the solution to (7) as μ_{opt}.

Deriving μ_{opt} analytically is not straightforward. Therefore, we estimate it numerically using the following simple algorithm.[1]

Result: μ_{opt}
initialization: $Mu = [\mu_1, \mu_2, ..\mu_L]$, $M_{opt} = []$
for $l \leftarrow$ *1:L* **do**
  $\rho(l) = \frac{\lambda}{Mu(l)}$
  $N_{packet}(l) = \frac{1-\rho(l)}{1-\rho(l)^{K+1}} \sum_{k=0}^{K} k\rho(l)^k$
  $T_w(l) = \frac{N_{packet}(l)}{\lambda}$
  if $lat_{max} - T_w(l) \geq \epsilon$ **then**
    M_{opt}.append($Mu(l)$)
  else
    reject $Mu(l)$
  end if
end
$\mu_{opt} = \min M_{opt}$

Algorithm 1: Calculation of μ_{opt} that allows to respect the latency requirement of a URLLC slice.

The steps of this procedure are as follows: First, we generate L candidate values for μ and keep them in a vector Mu. The number of values to generate is limited: For example, since $dpRB$ is assumed for now fixed, we can generate one μ value for each possible number of pRBs, which is defined by the available bandwidth for the given radio technology (e.g, for a bandwidth of 5Mhz, a maximum of 25 pRBs can be used) using (2). Then, we calculate N_{packet} corresponding to each value of μ and the resulting T_w value, which we compare with Lat_{max} to check if condition (6) is respected.

Note that we use a latency margin ϵ when we compare T_w with Lat_{max} to accept or reject a μ value. By appropriately controlling ϵ, we can ensure that T_w is adequately lower than the latency threshold Lat_{max}, but also close enough to it in order not to waste a lot of resources, while respecting condition (6).

Out of all the μ values that lead to an acceptable latency (in case there are multiple), we select as the optimal the one which minimizes the difference between T_w and Lat_{max}, i.e., the smallest value of M_{opt}. These steps are illustrated in Algorithm 1.

Once μ_{opt} is obtained, we use (8) to derive the corresponding N_{pRB} to be assigned to a URLLC slice. As for the case of eMBB, the proposed method needs to be run for each eNodeB where UEs of the slice are connected to.

[1] Adaptations of standard numerical techniques such as the Newton-Raphson and the bisection algorithms are also applicable.

IV. A CHANNEL QUALITY-DRIVEN ALGORITHM FOR DYNAMIC N_{pRB} ESTIMATION

The initial number of resource blocks calculated per slice in the previous step is based on the assumption that d_{pRB} is fixed for all users. However, users experience different channel conditions, and hence different data rates.

Therefore, we propose to correct the estimation of the d_{pRB} by using per-UE channel quality reports obtained from eNodeBs. These reports include the CQI and MCS values of each UE belonging to a cell. Note that these values are transmitted to eNodeBs by the UEs in order to be used in the scheduling process. We organize these CQI values in a matrix $v(j,k)$, where j is the id of the slice and k is the id of the UE. Based on the CQI, we can estimate d_{pRB} per UE and per cell (eNodeB). Indeed, d_{pRB} can be obtained by using the same tables used by the eNodeB to translate a CQI to a data rate [10]. Matrix v is then transformed to a matrix of data rates noted $d_{pRB}(j,k)$, where j and k have the same meaning as for matrix v.

Algorithm 2 presents the different steps of the dynamic slice resource allocation procedure. We note that $Slice(j)$ gives the type of the deployed slice, $N_{pRBopt}(i,j)$ is a matrix that gives for each cell i the necessary number of $pRBs$ for slice j, $N_{users}(i,j)$ a vector indicating the number of users of a slice j in cell i, and $d_{App\_user}(j)$ the data rate required by an application (per user) running on top of a slice j. This algorithm allows to estimate the N_{pRB} allocated to each slice and for each network cell more accurately: For an eMBB slice, it sums the necessary resources per UE considering each user's individual radio capacity reflected in $d_{pRB}(j,k)$. For URLLC, it applies (8), using the optimal service rate as computed by Algorithm 1 to attain latency requirements, and the mean achievable d_{pRB} across all slice users per eNodeB considering each user's channel quality, instead of a fixed optimistic value for all.

Note that this algorithm is run periodically by the SO. It relies on the eNodeBs' reports obtained also periodically. The periodicity of running these algorithms is independent from the scheduling period TTI used at the MAC layer of the eNodeBs.

Result: $N_{pRBopt}(i,j)$
for *each cell* i **do**
  for *each slice* j **do**
    if Slice (j) == eMBB **then**
      $N_{pRBopt}(i,j) = \sum_{k=1}^{k=N_{users}(i,j)} \frac{d_{App/user}(j,k)}{d_{pRB}(j,k)}$
    else
      if Slice (j) == uRLLC **then**
        $N_{pRBopt}(i,j) = \frac{\mu_{opt} * avg\_packet\_size}{\frac{1}{N_{users}(i,j)} \sum_{k=1}^{k=N_{users}(i,j)} d_{pRB}(j,k)}$
      end if
    end if
  end
end

Algorithm 2: Calculation of N_{pRB} for eMBB and uRLLC slices for multiple cells

V. PERFORMANCE EVALUATION

A. Scenarios and parameters

To evaluate the performance of the proposed solution, we extended the Matlab implementation of the two-level scheduler used in [1]. We mainly modified the SO part to include our algorithms. In this simulation, we considered two types of slices, i.e., eMBB and URLLC. Each slice is defined by the required application data rate, the number of users, the maximum latency for URLLC, etc.

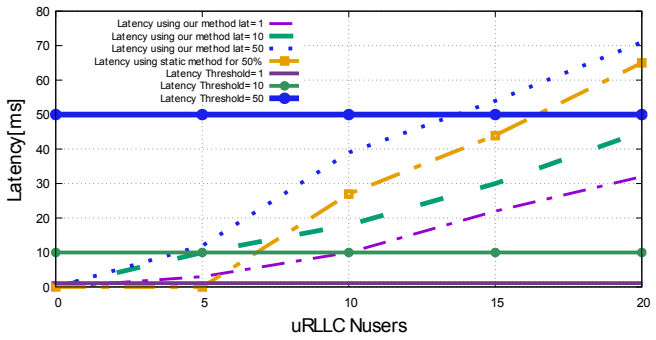
We simulated different scenarios, where we varied the number of users of the URLLC slice ($N_{uRLLCusers}$) while keeping it fixed for the eMBB slice ($N_{eMBBusers}$) to 5 users, and for different channel qualities: (i) medium quality where the CQI varies from 7 to 9; (ii) good quality where the CQI varies from 13 to 15. The different channel qualities will directly affect $dpRB$, which allows us to see its impact on the proposed solutions. Note that we simulated the case of only one eNodeB and one SO. Table I presents the simulation parameter set in all scenarios:

TABLE I: Simulation parameters

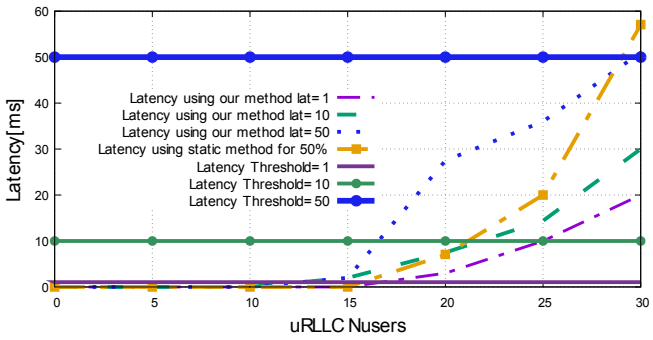
Parameter	Values
Slices	[uRLLC, eMBB]
Average Packet Size	[20, 125] bytes
Data rate	[160, 1000] kbit/s
TTI	[1, 1] ms
SO interval	1 s

We compared our solution with the one adopted in [1], which shares the $pRBs$ among the different slices using a statically selected percentage; in our tests we considered a 50% slice-dedicated bandwidth (SDB) per slice. It is worth noting that the number of available $pRBs$ is bounded by the channel bandwidth. For our simulation, we used a channel bandwidth of 5Mhz, where 25 $pRBs$ are available. We selected this number to saturate quickly the channel and show the efficiency of our solution. For higher bandwidths, the only difference concerns the threshold from where our solution does not perform well. It may happen that the combined number of pRBs to be allocated to both eMBB and uRLLC exceeds the channel capacity; hence, we adopted in this implementation a fair share of the resources, which has been computed as follows. First we compute $\Delta = N_{pRBmax} - (N_{pRBuRLLC} + N_{pRBeMBB})$ that represents the difference between the available number of pRBs and the requested number of pRBs for both slices. Then, we reduce the same amount of pRB ($\frac{|\Delta|}{2}$) from each slice, in order to fit the capacity of the channel. Other policies could be used, such as giving high priority to one slice, by first satisfying this slice and giving the remaining pRBs to the other slice. In this paper we use only the fair share of the channel, leaving other policies for future work.

Finally, we computed three main metrics: the eMBB slice throughput, the URLLC latency and the variation of N_{pRB} for each slice. We varied the number of URLLC slice users from 1 to 20 in the case of the medium-quality channel, and from 1 to 30 in the case of the good-quality channel, while fixing the number of eMBB users to 5. The presented results are averaged after several runs of the simulation.

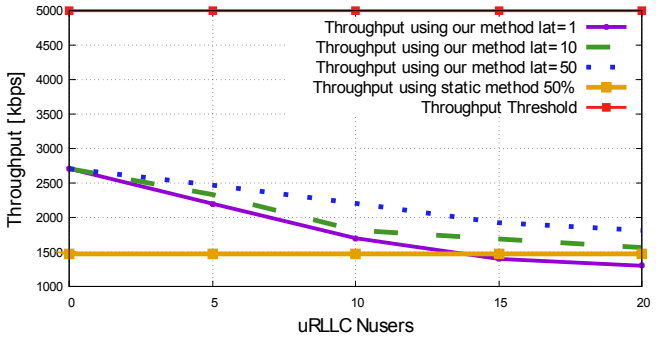


(a) Medium channel quality.

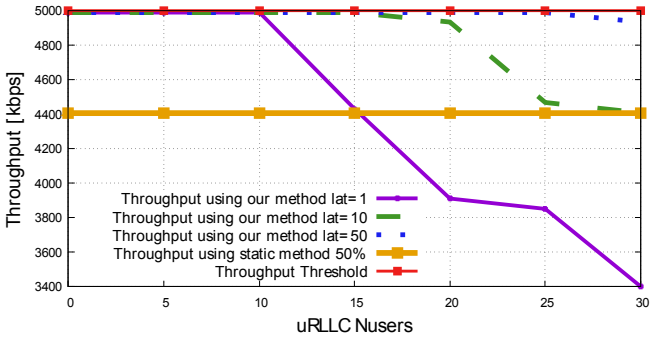


(b) Good channel quality.

Fig. 1: Latency vs. the number of URLLC users.



(a) Medium channel quality.



(b) Good channel quality.

Fig. 2: Throughput vs. the number of URLLC users.

It is worth noting that our main objective is to evaluate the accuracy of our proposed methods to well estimate the needed radio resources for each type of slice.

B. Results

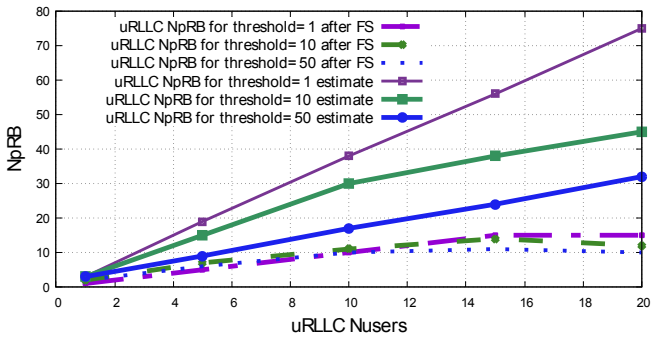
Fig. 1a and 1b illustrate the latency experienced by the uRLLC users for different numbers of N_{users_uRLLC}, and for two channel qualities, good and medium.

Here we considered different values for Lat_{max}: 1 ms, 10 ms and 50 ms, which reflect different service-level requirements. We remark that our algorithm allows to keep the latency around Lat_{max}, whatever the value of the latter and for both channel qualities. However, we see that there is a threshold (i.e., number of URLLC users) beyond which latency exceeds Lat_{max}; 2, 5 and 14 for the medium channel quality for Lat_{max}=1 ms, 5 ms and 50 ms respectively, and 15, 22 and 29 for the good channel quality for Lat_{max}=1 ms, 5 ms and 50 ms respectively. The difference between these values is explained by the fact that good channel quality permits to have higher N_{pRB} compared with the medium channel quality, thus accommodating more URLLC users. In addition, we observe that using a fixed number of pRBs cannot guarantee the very low latency requirement, as the used value (i.e., 50%) is not optimal (see Fig. 1a and 1b).

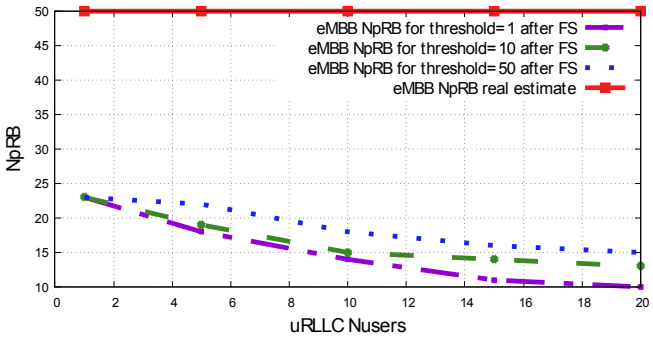
Fig. 2a and 2b show the throughput obtained for the eMBB slice as a function of the number of the URLLC slice's users. We remark the same behaviour as in the precedent figures. Namely, there is a threshold beyond which the performance of the slice degrades, particularly in the case of a good channel quality. Indeed, for the medium channel quality, our solution cannot guarantee the requested bandwidth (5 users $\times$ 1 Mbps). However, for the good channel quality our solution guarantees the needed bandwidth until 10 and 25 users when Lat_{max}=1 ms and 50 ms, respectively. This is expected, as in the case of Lat_{max}=1 ms the URLLC users need more pRBs, which strongly affects the eMBB users (see Figures 3b and 4b). Regarding the static assignment of pRBs, it ensures always the same throughput (lower than 5mbps), which is not optimal.

To better understand the obtained results, we have drawn in Fig. 3 and 4 the number of pRBs ($NpRB$) estimated and used by the eNodeBs for each type of slice, and for both channel qualities. From Fig. 3a and 4a we clearly see that the estimated value of N_{pRB} is similar to the one communicated to the eNodeB, until reaching the identified thresholds in Fig. 1a and 1b. When exceeding these thresholds, the communicated N_{pRB} to eNodeB are lower than the estimated value. This is mainly because the channel capacity is exceeded, and the proposed solution starts using the fair share of pRBs among the two slices. Hence, it is possible to accommodate the URLLC requirement for both channel qualities.

Regarding the eMBB slice, where the results are displayed in Fig. 3b and 4b for both channel qualities, the estimated N_{pRB} cannot be satisfied in case of medium channel quality,

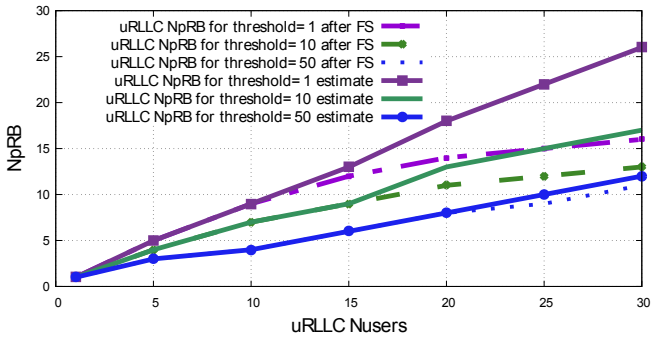


(a) NpRB of uRLLC slice.

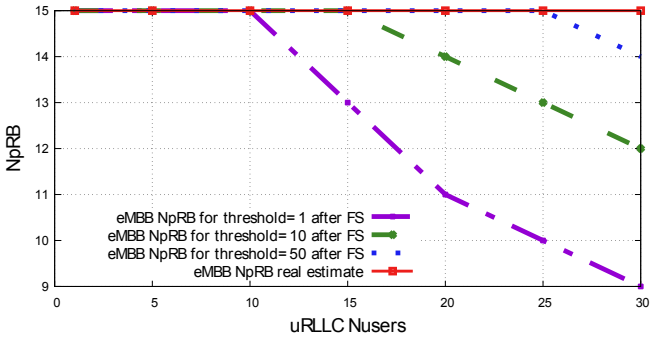


(b) NpRB of eMBB slice.

Fig. 3: Number of pRBs vs. the number of URLLC users for a medium channel quality.



(a) NpRB of uRLLC slice.



(b) NpRB of eMBB slice.

Fig. 4: Number of pRBs vs. the number of URLLC users for a good channel quality.

and after exceeding the identified threshold in Fig. 2a and 2b for good channel quality. Furthermore, we remark that the number of needed N_{pRB} is higher in case of medium channel quality, which is expected as $dpRB$ in this case is lower than when channel quality is better; hence more pRBs are needed to satisfy the throughput of eMBB users.

Overall, these results confirm that the proposed model to estimate the needed N_{pRB} for eMBB and URLLC employed by our proposed solution is accurate and permits to solve the problem of sharing the RAN resources among slices.

VI. Conclusion

In this paper, we addressed the problem of slicing and isolating RAN resources in slicing-ready 5G networks using the concept of two-level scheduling introduced in [1]. We proposed two algorithms that estimate the needed RAN resources for two types of 5G slices: eMBB and uRLLC. We used simulation to evaluate the performance of the proposed algorithms under different channel conditions. The obtained results allowed us to verify the accuracy of our algorithms when estimating the needed pRBs for each type of slice. The proposed algorithms are used at the slice orchestrator level and could be easily implemented in a real platform. Our future work will focus on improving the estimation of the URLLC algorithm by relaxing the assumptions on the traffic characteristics, using more general models.

VII. Acknowledgement

This work was partially supported by the European Union's Horizon 2020 Research and Innovation Program under the 5G!Drones (Grant No. 857031) and 5G-TRANSFORMER (Grant No. 761536) projects.

References

[1] A. Ksentini et al., "Providing low latency guarantees for slicing-ready 5G systems via two-level MAC scheduling," in *IEEE Network*, Nov. 2018.

[2] A. Ksentini and N. Nikaein, "Towards enforcing Network Slicing on RAN: Flexibility and Resources abstraction," in *IEEE Communications Magazine*, Jun. 2017.

[3] X. Foukas et al., "FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks," in *Proc. ACM CoNEXT*, 2016.

[4] P. Caballero, "Multi-Tenant Radio Access Network Slicing: Statistical Multiplexing of Spatial Loads," in *IEEE/ACM Transactions on Networking*, vol. 25(5), Oct. 2017.

[5] A. Anand et al., "Joint Scheduling of uRLLC and eMBB Traffic in 5G Wireless Networks," in *Proc. IEEE INFOCOM*, 2018.

[6] P. Caballero et al., "Network Slicing for Guaranteed Rate Services: Admission Control and Resource Allocation Games," in *IEEE Transactions on Wireless Communications*, vol. 17(10), Oct. 2018.

[7] V. Sciancalepore et al., "Mobile Traffic Forecasting for Maximizing 5G Network Slicing Resource Utilization," in *Proc. IEEE INFOCOM*, 2017.

[8] D. Bega et al., "CARES: Computation-aware Scheduling in Virtualized Radio Access Networks," in *IEEE Transactions on Wireless Communications*, vol. 17(12), 2018.

[9] J. Salvat et al., "Overbooking Network Slices through Yield-driven End-to-End Orchestration," in *Proc. ACM CoNEXT*, 2018.

[10] *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures*, 3GPP TS 36.213, v. 15.2.0, Release 15, Oct. 2018.