Slice Scheduling with QoS-Guarantee towards 5G

Robert Schmidt, Chia-Yu Chang, Navid Nikaein
Communication Systems Department, EURECOM, France
Email: {robert.schmidt, chia-yu.chang, navid.nikaein}@eurecom.fr

***Abstract*—Future mobile networks are supposed to handle a variety of services with different requirements. Network slicing is considered to be a key enabler to cope with the increasing complexity of these networks. This includes slicing of the radio resources in order to use them efficiently. In this paper, we propose a radio resource slicing system for three types of slices with specific radio resource/quality of service (QoS) requirements. It enables co-existence of (1) rate-based/efficiency-oriented and (2) low-latency slices, as well as (3) slices with fixed allocations. Slice scheduling is based on utility functions with a priority-based resource allocation. Using simulations, we validate the applicability of the proposed system, and demonstrate that both a guaranteed throughput and low delay for different slices at the same time is possible. Our system outperforms existing slicing solutions in terms of delay requirement satisfaction and efficient resource utilization.**

I. Introduction

Recent years have witnessed a trend towards a service-oriented fifth generation (5G) mobile network. The International Telecommunication Union (ITU) has identified some key usage scenarios alongside respective requirements in ITU-R M.2083-0 to be fulfilled. Besides the enhanced data rate for human-centric use cases, summarized as enhanced mobile broadband (eMBB), more device-centric use cases, like ultra-reliable and low latency communications (URLLC) and massive machine type communications (mMTC), will reshape the new 5G services in terms of latency and connection density. These services have partially conflicting objectives, and abstractions are needed to efficiently accommodate them within the same network infrastructure while coping with increasing complexity. This led to the proposal of network slicing (NGMN Alliance, "Description of network slicing concept") to create separated virtual spaces for multiple services to flexibly and efficiently use the network.

More specifically in the radio access network (RAN) domain, several virtual base stations (BSs) are built for each service with individual state, processing and resource [1] on top of the underlying BS. Among these three elements, radio resources are the most scarce, and thus a slicing system has to strike the balance among different objectives, such as using the spectrum efficiently for many users while guaranteeing delivery for high-reliability users. In addition, a slicing system shall guarantee that (i) changes in one service do not affect the system and/or violate the service-level agreement (SLA) of other services, and (ii) redistribute unused resources to maximize resource efficiency.

As the origination of RAN slicing, network slicing considers a collection of logical overlay networks over a physical network [2]. Also, any action taken by one slice will not negatively affect other slices [3], which is an ideal match for the vision of multiple services in 5G. In practice, several works study the RAN slicing in terms of resource allocation. In [4], a proposed RAN slicing architecture allows radio resource management policies to be enforced at the level of physical resource blocks (PRBs) by a resource visor towards each slice. Orion [5] introduces the BS hypervisor to share the radio resources in terms of virtual resource blocks (vRBs) to be grouped and provided to the corresponding slice. The resource abstraction approach in [6] can support three types of resource requirements: PRB, vRB and virtual transport block. The NVS approach [7] can slice the RAN domain resources for two types of resource requirements: data rate and vRBs. Nevertheless, none of above works consider the impacts of quality of service (QoS) on the RAN domain resources for three key types of resource requirements: physical resources, throughput, and latency.

This paper proposes a RAN slice scheduling framework with QoS guarantee to meet the performance requirements of different usage scenarios as identified by ITU. It introduces three types of slices with their associated formulated utility functions as: (a) *fixed* slices requesting dedicated physical resources, (b) *dynamic* slices asking for aggregate throughput like eMBB, and (c) *on-demand* slices demanding latency requirement like URLLC. Finally, it achieves QoS guarantee by applying a priority-based resource allocation tailored to each type of slice.

This paper is organized as follows. Section II outlines the considered multi-service slicing approach and then highlights some key challenges. In Section III, we present the proposed slicing system and formulate the corresponding utility function for each type of service and the associated three-step weight-based radio resource scheduler with QoS awareness. Finally, Section IV provides simulation results and reveals the effectiveness of the proposed framework in terms of guaranteeing QoS requirements.

II. Slice Representation and Challenges

A. Overview

In this work, we focus on the RAN domain, comprising several BSs, to be sliced for multiple network services. In practice, a slice descriptor will serve as a blueprint to craft a slice instance into the virtual base stations (vBSs), which is abstracted and embedded into the underlying BS and then exposed through the controller interface [1]. On one hand, the slice descriptor formally addresses the requirements from the

slice owner's perspective, such as the required guaranteed bit rate (GBR), to be applied to the corresponding vBS. On the other hand, from an operator's point of view, the objective is to embed as many vBSs as possible into the underlying BS, while satisfying their service requirements. To this end, a set of rules can be applied to the embedding operation to pursue these two goals simultaneously.

To fully represent a BS, we resort to a triple [1] consisting of (i) resources[1], (ii) processing[2], and (iii) state[3]. This triple within a vBS can be inferred from the SLA within the slice descriptor. For instance, the SLA can comprise the required resource parameters (e.g., the GBR or maximum delay threshold), processing parameters (e.g., isolated CP processing), and configuration parameters (e.g., customized admission control rules). According to these contents within the slice descriptor, the controller can perform admission control for the resource parameters and checks the feasibility for specified processing and configuration parameters. Finally, the vBS is created to reveal a slice-specific view of the BS to the slice owner.

B. Challenges of embedding vBS into BS

Within the aforementioned process overview, a number of challenges emerge during the procedure of embedding multiple vBSs into a single BS. Especially, most of the attention is given in the resource aspect as highlighted in Section I, and thus we highlight three major challenges as follows:

1) **Requested resource isolation**: For instance, emergency services will request dedicated radio resource to ensure complete isolation from others.
2) **Efficient resource utilization**: Taking the eMBB service as an example, it aims to pursue efficient resource utilization with loose requirements on delay and isolation.
3) **Performance guarantee**: The URLLC service will request hard delay guarantees that need to be maintained with sub-optimal resource utilization efficiency.

To better depict these challenges of vBS embedding, we identify three corresponding slice resource requirement types, which will be elaborated in the following section.

III. QoS-aware Resource Slicing Framework

In this section, we first identify three types of resource requirements, then formulate the corresponding utility functions, and finally propose the weight-based QoS-aware resource slicing framework for the vBS embedding operation.

A. Three types of slice resource requirements

In particular, three types of slice resource requirements are of interest in this work:

- A *fixed* slice requests dedicated radio resources along time and frequency domains. Therefore, it is isolated from other slices and does not multiplex its resources with others. One example is the bandwidth parts (BWPs) defined in 5G (see 3GPP TS 38.211) that operate on disjoint parts of the spectrum with a given numerology.
- A *dynamic* slice requests a share of resources in terms of aggregate throughput. It can be mapped to the eMBB usage scenario, and thus a precise radio resource allocation is less important than the efficient use of available resources for a guaranteed throughput.
- An *on-demand* slice exhibits more stringent requirements in terms of latency, and hence it can be mapped to the URLLC scenario. Therefore, such slice should be assigned radio resources with short delay, in comparison with the dynamic slice.

Based on these slice resource requirements, we propose respective utility functions from which we will introduce a novel slice scheduling algorithm with QoS awareness.

B. Utility functions for different types of slice

Corresponding to the three types of slice resource requirements, we form three slice sets grouping the slices with respective types of resource requirements: $\mathcal{K}_{\text{fix}}$, $\mathcal{K}_{\text{dyn}}$, and $\mathcal{K}_{\text{ond}}$. As for the k-th slice, its utility function is represented either as $\mathcal{U}_k$, $\mathcal{V}_k$, or $\mathcal{W}_k$, if $k \in \mathcal{K}_{\text{fix}}$, $k \in \mathcal{K}_{\text{dyn}}$ or $k \in \mathcal{K}_{\text{ond}}$, respectively. From the perspective of the operator, it aims to maximize the summation of utility functions from all instantiated slices, while still guaranteeing QoS requirements:

$$\mathcal{U} = \sum_{k \in \mathcal{K}_{\text{fix}}} \mathcal{U}_k + \sum_{k \in \mathcal{K}_{\text{dyn}}} \mathcal{V}_k + \sum_{k \in \mathcal{K}_{\text{ond}}} \mathcal{W}_k. \tag{1}$$

1) Fixed slices: For a fixed slice $k \in \mathcal{K}_{\text{fix}}$, its SLA stipulates that such slice requests P_k RBs in every scheduling interval, starting at RB P_k^{start}. To ensure that no overlapping between fixed slices occurs, extra admission control is required and will be elaborated in Section III-C. Then, the slice's utility function can be written as

$$\mathcal{U}_k = \begin{cases} 1 & \text{if RB range } [P_k^{\text{start}}, P_k^{\text{start}} + P_k) \text{ allocated} \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

i.e., such slice can only be satisfied when the exact requested resources are allocated, as required in Section II-B.

2) Dynamic slices: We modify the formulation of bandwidth-based slices in [7] to enable concurrent slice scheduling (multiple slices scheduled within the same subframe/slot). The SLA of a dynamic slice $k \in \mathcal{K}_{\text{dyn}}$ requires a requested rate R_k as well as a prioritization rate R_k^{ref}, both in bps. The former describes the rate that the slice shall achieve in the long run. It is measured through the slice's achieved cumulative rate r_k in bps, and should fulfill $r_k \geq R_k$. The prioritization rate serves as a reference which this slice should achieve within the scheduled RBs to maintain its resource efficiency. It is checked through the average per-RB rate $\bar{r}_k$.

For a BS with N RBs, we control the attempted allocation rate R'_k which considers the resource efficiency via dividing

[1]It describes the radio spectrum resources, consisting of bands, carriers, or available resource blocks (RBs).

[2]It refers to the functional blocks to perform the necessary control plane (CP)/user plane (UP) operations, delimited through functional splits.

[3]It refers to the status of CP/UP processing and the associated configuration.

the estimated allocation rate $\bar{R}_k = N \cdot \bar{r}_k$ by the prioritization rate R_k^{ref}:

$$R'_k = R_k \cdot \min\left(1, \frac{\bar{R}_k}{R_k^{\text{ref}}}\right) = R_k \cdot \min\left(1, \frac{N \cdot \bar{r}_k}{R_k^{\text{ref}}}\right) . \quad (3)$$

Note that if $\bar{R}_k$ is above the reference rate R_k^{ref}, the requested rate R_k is targeted. Otherwise, the requested rate is proportionally scaled down to get the attempted allocation rate R'_k. The rationale behind R'_k is to increase the resource efficiency, as tailored in the second challenge of Section II-B.

Finally, to achieve proportional fairness among different dynamic slices, the utility function can be expressed as

$$\mathcal{V}_k(r_k) = \frac{R'_k}{N \cdot \bar{r}_k} \log(r_k) . \quad (4)$$

3) On-demand slices: The SLA of an on-demand slice $k \in \mathcal{K}_{\text{ond}}$ includes a delay-threshold Δ_k in milliseconds, a packet-loss probability δ_k, and a long-run maximum resource share $t_{k,\max}$ in terms of the ratio of RBs (i.e., $0 \leq t_{k,\max} \leq 1$). Beyond the aforementioned SLA, for each slice scheduling interval, a slice signals the number of bits B_k to be scheduled to satisfy its delay-threshold Δ_k. Based on the above parameters (i.e., B_k and $t_{k,max}$) as well as measured statistics (e.g., channel quality, user number), an on-demand slice can estimate its delay bound of the scheduling process.

To maintain the above SLA, the operator measures the achieved resource share t_k as a fraction of the N RBs, and the average per-RB transportation bit rate $\bar{b}_k$.

Finally, we can write the utility function using the similar approach for dynamic slices as:

$$\mathcal{W}_k(t_k) = \frac{B_k}{N \cdot \bar{b}_k} \log(t_k) = T_k \log(t_k), \quad (5)$$

in which $T_k = \frac{B_k}{N \cdot \bar{b}_k}$ represents the requested resource[4]. It is subject to $t_{k,\max}$ as elaborated in Section III-C.

Since we follow the same formulation approach as for the dynamic slice, (4) and (5) show several similarities. Such observation is also found in [7], and the maximization over these two utility functions can be shown to be equivalent[5].

C. Proposed multi-slice scheduler with QoS-awareness

To utilize above utility functions, we hereby propose a multi-slice scheduler comprising three major steps:

1) Based on the SLA of fixed slices, we first allocate the dedicated requested resources.
2) Then, following the QoS requirements of on-demand slices, we allocate necessary resources.
3) Finally, we schedule resources for dynamic slices.

Nevertheless, as mentioned in Section III-B, extra *admission control* is required prior to the creation of slices in order to ensure that the resource usage of all slices can be reasonably accommodated. Specifically, two constraints are applied. In the first constraint, we restrict that available N RBs can sustain the resource requirements from all slice instances. Thus, we can write the ratio of requested resources from fixed slices as $\frac{P_k}{N}, \forall k \in \mathcal{K}_{\text{fix}}$, the allocated resource ratio for on-demand slice as $t_k, \forall k \in \mathcal{K}_{\text{ond}}$, and the allocated resource ratio for dynamic slices as $\frac{r_k}{R_k}, \forall k \in \mathcal{K}_{\text{dyn}}$. Assuming that on-demand slices use their agreed maximum resource share $t_k = t_{k,\max}$ and dynamic slices the attempted rate $r_k = R'_k$, we have

$$\sum_{k \in \mathcal{K}_{\text{fix}}} \frac{P_k}{N} + \sum_{k \in \mathcal{K}_{\text{ond}}} t_{k,\max} + \sum_{k \in \mathcal{K}_{\text{dyn}}} \frac{R'_k}{\bar{R}_k}$$
$$\leq \sum_{k \in \mathcal{K}_{\text{fix}}} \frac{P_k}{N} + \sum_{k \in \mathcal{K}_{\text{ond}}} t_{k,\max} + \sum_{k \in \mathcal{K}_{\text{dyn}}} \frac{R_k}{R_k^{\text{ref}}} \leq 1 \quad (6)$$

where the first inequality follows from (3). This gives a sufficient condition on the slices' SLAs. In the second constraint, admission control needs to ensure that no fixed slice is overlapping with other fixed slices. By denoting $[P_k^{\text{start}}, P_k^{\text{start}} + P_k)$ as the range of RBs allocated to the fixed slice $k \in \mathcal{K}_{\text{fix}}$, we can have the following restriction:

$$[P_i^{\text{start}}, P_i^{\text{start}} + P_i) \cap [P_j^{\text{start}}, P_j^{\text{start}} + P_j) = \varnothing, \forall i \neq j . \quad (7)$$

After admission control, we follow the aforementioned three steps for the scheduling operation. Note that we adapt a weight-based approach in each step, i.e., the weight for the k-th slice is w_k, to maximize the marginal utility of different types of service requirements.

1) Fixed slices: As for the fixed slice, we can directly follow the resource requirements within the SLA, as shown in Algorithm 1. Note that the weights for all fixed slices are set as $w_k \leftarrow \infty, \forall k \in \mathcal{K}_{\text{fix}}$, since they are guaranteed to be satisfied after applying (7) in admission control.

Algorithm 1: Allocation of RBs for fixed slices.

Input : $\text{SLA}_k = \{P_k^{\text{start}}, P_k\}, \forall k \in \mathcal{K}_{\text{fix}}$
Output: Allocation results for all fixed slices
forall $k \in \mathcal{K}_{fix}$ **do**
  Allocate P_k RBs starting from P_k^{start} to slice k
end

2) On-demand slices: Before scheduling resources for on-demand slices, we first derive the weights w'_k based on the marginal utility via a derivation approach:

$$w'_k = \frac{d}{dt_k} \mathcal{W}_k(t_k) = \frac{T_k}{t_k} = \frac{B_k}{b_k} \quad (8)$$

in which $b_k = N \cdot \bar{b}_k \cdot t_k$ is the average transported bits during each scheduling interval for slice k. However, this weight ignores the QoS requirements between slices required in Section II-A. Therefore, we can apply a similar approach as the M-LWDF scheduler [8] and multiply this weight with the QoS-related coefficients, i.e., the delay threshold Δ_k, the packet loss probability δ_k, and the delay reported by the slice D_k. Finally, the weight for slice k can be represented as:

$$w_k = -\frac{\log \delta_k}{\Delta_k} \cdot D_k \cdot w'_k = -\frac{\log \delta_k}{\Delta_k} \cdot D_k \cdot \frac{B_k}{b_k} , \quad (9)$$

[4]Since the resource efficiency is not targeted by on-demand slices, the down-scaling operation of (3) is not applied here.

[5]Due to space reasons, we encourage the readers to refer to Lemma 1 of [7].

which will be used by our proposed algorithm to determine the scheduling ordering for on-demand slices. The average rate b_k is approximated through an exponential moving average b_k^{exp} at the scheduling interval j:

$$b_{k,j}^{\text{exp}} = (1-\beta) \cdot b_{k,j-1}^{\text{exp}} + \beta \cdot b_{k,j-1}^{\text{inst}} \,, \tag{10}$$

where $b_{k,j-1}^{\text{inst}}$ is the achieved rate at scheduling interval $j-1$, and β is a small positive constant.

Afterwards, we need to determine the amount of RBs to be allocated for each on-demand slice. A naive approach would be to limit each slice to its maximum share of resources of $t_{k,\max}$. However, this would penalize slices that did not request resources in the past, and we therefore resort to fulfill the maximum resource share over a moving time window. In this sense, more than $t_{k,\max}$ resources can be scheduled in a single scheduling interval, if few or zero resources were requested before. To better represent such condition, we collect the historic resource usage ratio for the k-th slice at the j-th scheduling interval as $t_{k,j}$, and denote the average resource usage ratio also at the j-th scheduling interval as $\bar{t}_{k,j}$ with the following relations:

$$\bar{t}_{k,j} = \frac{1}{\tau_k} \sum_{i=0}^{\tau_k - 1} t_{k,j} \approx \left(1 - \frac{1}{\tau_k}\right) \bar{t}_{k,j-1} + \frac{1}{\tau_k} t_{k,j} \,. \tag{11}$$

Note that such average resource usage ratio is approximated through an exponential moving average via only taking the historic resource share within previous τ_k scheduling intervals. We derive the maximally allowed resource share within the j-th scheduling interval $t_{k,j,\max}$ as:

$$\begin{aligned} & \bar{t}_{k,j} \leq \left(1 - \frac{1}{\tau_k}\right) \bar{t}_{k,j-1} + \frac{1}{\tau_k} t_{k,j,\max} = t_{k,\max} \\ \Rightarrow \quad & t_{k,j,\max} = \tau_k \cdot t_{k,\max} + (1-\tau_k) \cdot \bar{t}_{k,j-1} \,. \end{aligned} \tag{12}$$

Finally, the assigned resources are limited by the maximally allowed resource share over time or the requested resources:

$$T'_{k,j} = \min(t_{k,j,\max}, T_k) \,. \tag{13}$$

In summary, as shown in Algorithm 2, on-demand slices are allocated in the order of their weight w_k from (9) with a maximum resource share $T'_{k,j}$ from (13) at the j-th scheduling interval. A slice is not scheduled if $B_k = 0$.

3) Dynamic slices: To efficiently schedule dynamic slices, we allocate the remaining RBs to the slice with the highest marginal utility among those that have data to send. Based on the utility function of the dynamic slice, we can deduce the scheduling weights w_k using the derivative of the utility function. Note that the achieved rate r_k is the product of average RB rate $\bar{r}_k$, the number of RB N, and the resource share t_k of this slice:

$$\begin{aligned} w_k = \frac{d}{dt_k} \mathcal{V}_k(r_k) &= \frac{R'_k}{N \cdot \bar{r}_k} \cdot \frac{d}{dt_k} \log(N \cdot \bar{r}_k \cdot t_k) \\ &= \frac{R'_k}{N \cdot \bar{r}_k} \cdot \frac{N \cdot \bar{r}_k}{N \cdot \bar{r}_k \cdot t_k} = \frac{R'_k}{r_k} \,. \end{aligned} \tag{14}$$

Algorithm 2: Allocation of RBs for on-demand slices.

Input : $B_k, D_k, \text{SLA}_k = \{t_{k,\max}, \Delta_k, \delta_k\}, b_k^{\text{exp}}, \forall k \in \mathcal{K}_{\text{fix}}$
Output: Allocation results for on-demand slices $\in \mathcal{K}_{\text{fix}}$
forall $k \in \mathcal{K}_{ond}$ **do**
  Calculate w_k as in (9)
  Calculate T'_k as in (13)
end
$\mathcal{K}_{\text{ond}}^{\text{sorted}} \leftarrow$ sort $\mathcal{K}_{\text{ond}}$ desc. by w_k, and skip if $B_k = 0$
forall $k \in \mathcal{K}_{ond}^{sorted}$ *and while RBs available* **do**
  Allocate round($T'_k \cdot N$) RBs to slice k
end

Algorithm 3: Allocation of RBs for dynamic slices.

Input : $\text{SLA}_k = \{R_k, R_k^{\text{ref}}\}, r_k^{\text{exp}}, \forall k \in \mathcal{K}_{\text{dyn}}$
Output: Allocation results for dynamic slices $\in \mathcal{K}_{\text{dyn}}$
forall $k \in \mathcal{K}_{dyn}$ **do**
  Update r_k^{exp} as in (15)
  Calculate w_k as in (14)
end
$k_{\max} \leftarrow \arg\max_k(w_k)$
Allocate all remaining RBs to slice $k_{\max}$

Finally, as shown in Algorithm 3, the dynamic slice with the highest weight w_k from (14) is allocated, where the achieved rate r_k in (6) can be approximated through an exponential moving average $r_{k,j}^{\text{exp}}$ for slice k in (15):

$$r_{k,j}^{\text{exp}} = (1-\beta) \cdot r_{k,j-1}^{\text{exp}} + \beta \cdot r_{k,j-1}^{\text{inst}} \,. \tag{15}$$

IV. Simulation Results

We implemented the proposed algorithm in a 3GPP LTE-aligned simulator, written in Matlab, with the system parameters as described in 3GPP TS 36.212.[6] The simulations are done for a bandwidth of 5MHz (25 RBs) in a single antenna FDD mode. We consider a set of slices with a varying number of users. The scheduling is performed in two levels every subframe: (1) inter-slice scheduling with the proposed framework, and (2) intra-slice scheduling with a customized policy tailored to a slice requirement. In this paper, we focus on the downlink (DL) direction with variable wide-band channel qualities allowing dynamic modulation and coding scheme (MCS) per user and per subframe. Also, traffic profiles are defined on a per-user basis.

We first analyze the main slice requirements reusing the same methodology as defined in NVS [7], and assess the results for a 3GPP LTE system. Then, we evaluate scenarios in which more stringent QoS constraints and a strict isolation are necessary.

[6]The code is available at https://gitlab.eurecom.fr/schmidtr/mac-slice-sim upon request.

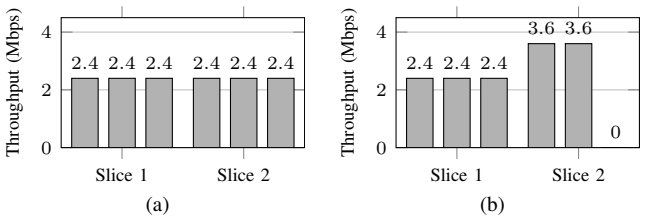

Figure 1. Isolation between two dynamic slices. (a) Initially, both slices have three users. (b) Then, one user in slice 2 disconnects, the slice increases per-user throughput while maintaining its aggregate throughput.

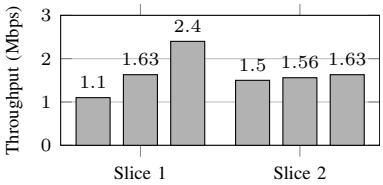

Figure 2. Customization through slicing: slice 1 operates with a proportional fair scheduler, slice 2 applies a blind equal throughput scheduler.

A. Main Slice Requirements

1) Isolation: We first show the isolation capabilities of the proposed slicing system. Consider two dynamic slices with the same SLA $R_k = 7.2\,\text{Mbps}$ and $R_k^{\text{ref}} = 14.4\,\text{Mbps}$. Each slice has three users with constant backlogged traffic. All users can have equal throughput as shown in Fig. 1a. After one of the users of slice 2 dissociates, we can observe in Fig. 1b that slice 2 maintains the aggregate throughput by redistributing its resources, without any impact on slice 1.

2) Customization: It should be possible to customize slices to achieve a differentiated treatment of user data flows. We perform the same experiment of two dynamic slices with different schedulers: Slice 1 uses a proportional fair scheduler [9] allocating resources proportionally to a user's current achievable throughput and its past average throughput. On the other hand, slice 2 employs a blind equal throughput scheduler [9] that strives for achieving the same average throughput, regardless of per-user channel quality. Both slices have three users with different channel qualities (MCSs 18, 23, and 28) and backlogged traffic. Fig. 2 shows that slice 2 achieves a lower aggregate throughput compared to slice 1 (4.6 Mbps instead of 5.1 Mbps), which confirms the customization property of the proposed framework and its potential impact of the resulting per-user and per-slice performance.

3) Efficient resource utilization: One of the main goals of resource slicing is an efficient resource utilization, and unused radio resources should be redistributed if allowed by the SLA. In this experiment, we consider two cases with (i) static slicing with three perfectly isolated slices and a reservation P_k of 28 %, 32 % and 40 % of bandwidth, similar to [4], against (ii) three dynamic slices that dynamically share their resources and a reservation R_k of 4.032 Mbps (corresponds to 20 %), 4.608 Mbps (32 %) and 5.76 Mbps (40 %), similar to [7]. Each slice has one user that periodically downloads a large file. As can be seen in Fig. 3a, resources are wasted when not all users are active. In case two (Fig. 3b), resources can be shared between slices. This can lead to increased throughput for active slices, and the cell usage is improved. Another effect is that the cell becomes completely unloaded, implying energy saving potential.

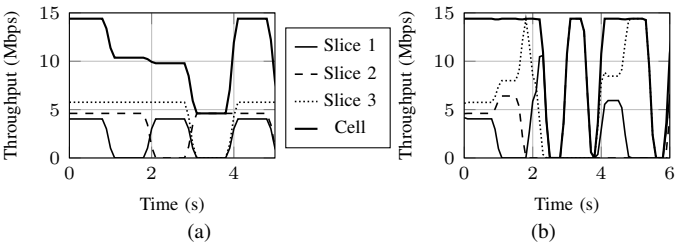

Figure 3. Exemplary resource utilization for three slices. (a) Static resource reservation. (b) Dynamic resource reservation.

B. Study of new scenario

1) Faster Delay Response: Within the prior work of NVS [7], there exist two critical problems. First, NVS "penalizes" lightly loaded slices, typical for URLLC, by scaling them down (cf. (3)) since it can not distinguish between bad channel quality and lightly loaded slices. Second, a slice has no control over *when* it is assigned resources. Thus, no delay bound can be formed. To overcome these problems, we introduced the on-demand slice in the proposed framework for which resources are allocated when needed.

We deploy three dynamic slices with each 4.0 Mbps requested throughput R_k (14.4 Mbps reference rate R_k^{ref}) and serving two users each with backlogged traffic. We compare the impact of the resource assignments with respect to QoS for (i) the original NVS slices (equivalent to our dynamic slice) to that of (ii) the on-demand slice. In case (i), we deploy an additional dynamic slice for QoS-sensitive traffic with a requested throughput R_k of 1.6 Mbps over a reference rate R_k^{ref} of 10 Mbps (corresponding to 16 %). In case (ii), we deploy an additional on-demand slice with a maximum resource share $t_{k,\max}$ of 16 %. The averaging window τ_k is set to 25 ms such that the slice can reuse previously unused resources.

Both additional slices serve four users with the same MCS 20. Each user requests packets of 255 B size and with 9 ms of allowed delay as for the 5G flow class 82 for delay-critical GBR traffic (see 3GPP TS 23.501). Packets arrive either in constant intervals of 10 ms and with 2 ms difference between the four users (denoted as CBR), or according to a Poisson Process with mean inter-arrival time 10 ms (corresponding to roughly 100 packets per user per second, denoted as VBR). We simulate 30 s.

Fig. 4a shows that the NVS-like slicing loses many packets for VBR traffic, since packets arriving grouped cannot be completely scheduled in the next scheduling window. The on-demand slices of the proposed framework, on the contrary, do not drop any packets at all. From Fig. 4b, it can also be observed that the on-demand slices use less resources as they request resources only when needed allowing efficient

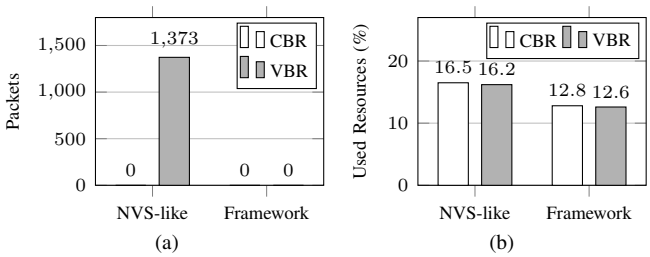

Figure 4. Comparison of (a) number of lost packets and (b) percentage of used resources over total system resources for 10 s of simulation.

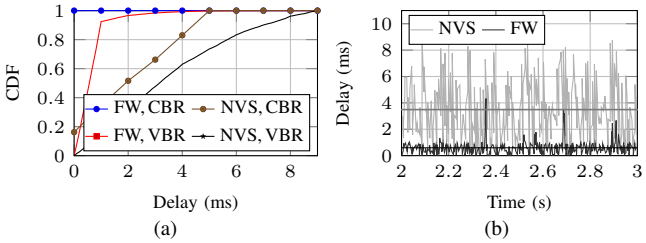

Figure 5. Comparison of delay of non-discarded packets between the proposed Framework (FW) and an NVS-like scheduling scheme (NVS): (a) CDF for CBR and VBR traffic. (b) Packet delays over 1 s simulated time for VBR traffic. Note the respective average delays, marked with horizontal lines.

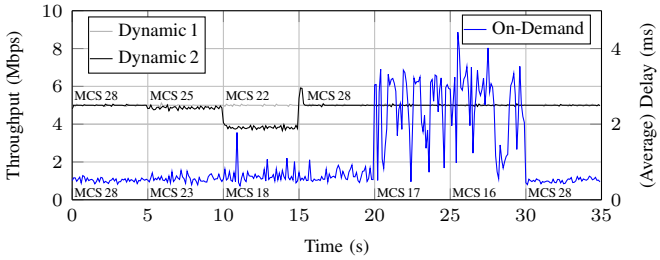

Figure 6. Behavior of dynamic (rate) and on-demand slices (delay) if user channel quality changes. The delay is an average over 100 ms.

resource utilization. On the contrary, NVS slices are assigned a complete subframe for transmission which might not always be possible to fill. This could also result in down-scaling of the NVS slice.

We report the packet scheduling delay as experienced by the users within a QoS-sensitive slice for non-dropped packets. Fig. 5a shows the cumulative distribution function (CDF) for the delay of packet transmissions. For CBR, our framework allows packet scheduling immediately after arrival, whereas NVS increases the delay to up to 5 ms. For VBR, we observe that our framework schedules packets reliably with 90 % of packets being scheduled before 1 ms and 99.9999 % before 8.0 ms[7] for the considered traffic scenario. NVS achieves the same percentile after 9 ms for VBR traffic by dropping packets. This is due to non-responsive behavior of NVS in which slices are scheduled purely with respect to their past throughput and not the actual queue state. Fig. 5b reports the delay for scheduled packets over a time duration of 1 s. Note how spikes in delay for on-demand slices (due to high traffic intensity) seldom reach the average delay of the NVS approach.

Finally, we note that the above traffic patterns favor NVS, since lower traffic would automatically result in packet loss due to down-scaling or resource limitation, subject to SLA. On the contrary, for on-demand slices, low traffic and after inactivity, a higher averaging duration τ_k in (12) can enable a more aggressive resource allocation.

In summary, we remark that the on-demand slice provides deterministic behavior subject to SLA when scheduling packets and thus greatly improves the latency and reliability of user data flows. Since the frame structure of 5G is highly similar, we expect analogous results if on-demand slices are scheduled on a slot basis (i.e., no mini-slot scheduling is employed).

2) Reaction to channel quality changes: As shown in Section IV-A1, individual dynamic slices are isolated from each other. Further, isolation of slices under changing channel qualities is ensured, as demonstrated in NVS [7].

In the following, we investigate the impact of changing channel qualities on a mix of dynamic and on-demand slices. We consider two dynamic slices with each 5.0 Mbps requested throughput R_k over a 12.5 Mbps reference rate R_k^{ref}. They each have two users with a constant 2.5 Mbps downlink bit rate and an initial MCS 28. At 5 s, the channel quality of users of one slice drops to MCS 25; at 10 s, to MCS 22; and at 15 s, MCS returns to 28 (cf. Fig 6). Furthermore, there is an on-demand slice with a maximum resource reservation $t_{k,\max}$ of 20 %. To increase the effect of increased delay due to decreased channel quality, we set the slice's resource averaging window τ_k in (12) to 10 ms. This slice contains four users and a packet arrival identical to the Poisson Process in Section IV-B1. The user channel quality, initially at MCS 28, drops to 23, 18, 17, and 16, at 5 s, 10 s, 15 s, and 25 s, respectively. It returns to MCS 28 at 30 s. The strong degradation of quality in the on-demand slice is motivated by the assumption that the slice still seeks to serve even bad users with high reliability requirements. The dynamic slice might be able to average out user quality drops over all its users, maintaining a high efficiency.

As shown in Fig. 6, the dynamic slice with diminishing user channel quality initially experiences instability in the throughput before dropping due to low channel quality. Note how in both cases, an adaptation is almost immediate. When the user channel quality reestablishes, the throughput for this slice initially overshoots for two reasons. First, since the rate adaptation needs less radio resources for the same amount of data, free resources can be used to catch up the previously low throughput. Second, the moving average $r_{k,j}^{\text{exp}}$ in (15) needs some time to adapt to the new, real throughput r_k, thus artificially boosting the weight of the dynamic slice in (14).

For the on-demand slice, Fig. 6 plots average delays over 100 ms windows. Down to MCS 18, a drop in channel quality results in a stronger variation of channel delays; the lower the quality, the higher the variation. Since the lower channel quality results in a lower rate for transmission, packet transmissions for some packets tend to take longer themselves and

[7]This duration includes all packets for the considered packet error rate, and the maximum data burst volume would exclude packet bursts that cannot be handled. See TS 23.501.

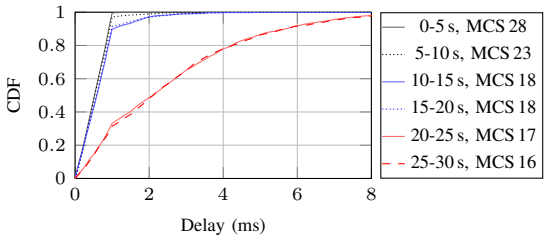

Figure 7. Comparison of scheduling delay for changing user channel qualities.

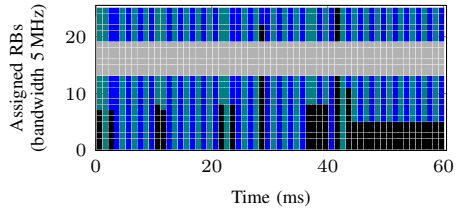

Figure 8. RB allocation for multiple slices: (light gray) fixed slice, (blue and green) two dynamic slices, (black) on-demand slice.

queuing up subsequent transmissions. Note that the quality of the dynamic slices has no impact on the on-demand slices since the on-demand slices are scheduled preferentially. This can also be seen in Fig. 7, which shows almost no difference in the CDF for the time windows 10-15 and 15-20 s. Starting at 20 s (Fig. 6) for qualities lower than MCS 18, a strong increase in latency (and a loss of packets) is noticeable for a slight decrease in MCS. It can be supposed that a sweet spot is reached for the parameter settings (arrival rate, resource share $t_{k,\max}$, windowing duration τ_k) such that the slice can frequently not request more than its maximum resource share to guarantee a short delivery time. For a fixed arrival rate, the latency could be reduced by setting a higher resource share $t_{k,\max}$ for this slice. However, this would constantly block resources in admission control and might create instability if overbooking is employed. Hence, it would be advisable to increase the windowing duration τ_k such that the scheduler might be temporarily granted more resources temporarily.

In summary, we note that on-demand slices have comparative, low scheduling delays down to a minimum channel quality for a given traffic pattern and parameter setting. Below this minimum quality, they show behavior similar to NVS-like or fixed slices (cf. Fig. 5a), since they are constantly restricted to their minimum resource share $t_{k,\max}$.

3) Behavior of on-demand slices: As already alluded in Section III-C, an on-demand slice is limited in the amount of RBs it can allocate to prevent that such a slice occupies all radio resources of the BS. In fact, a slice owner could fake the parameter B_k to be very high which the inter-slice scheduler uses without further checks. Hence we introduced the averaging parameter τ_k in (12) to limit an on-demand slice to its maximum allowed resource share $t_{k,\max}$, on average. This is shown in Fig. 8 where the on-demand slice uses almost all resources (at 30 ms), followed by no activity. However, as soon as it constantly requests resources, it is limited to its resource share $t_{k,\max} = 0.2$ (at 45 ms).

V. Conclusion

We presented a utility-based slice scheduling algorithm applicable to 4G and 5G systems to achieve QoS guarantees. It applies a priority-based resource allocation tailored to three types of slices, namely fixed slice, dynamic slice, and on-demand slice. Through extensive set of simulation, we demonstrated that the proposed framework maintains isolation, customization, and QoS guarantees, and significantly outperforms existing approaches, like NVS, in this respect.

Our framework is applicable to 5G since it uses a similar frame structure as 4G. However, 5G has additional means to accomodate low-latency services. In the future, we plan to evolve this work in the following directions: (i) considering 5G mini-slot scheduling within a slot, (ii) considering multiple BWPs with different numerologies, and (iii) integration with our FlexVRAN [1] and OpenAirInterface [10] platforms.

Acknowledgment

This work received funding from the European Union's Horizon 2020 Framework Programme under grant agreement No. 762057 (5G-PICTURE) and No. 761913 (SliceNet).

References

[1] R. Schmidt *et al.*, "FlexVRAN: A flexible controller for virtualized RAN over heterogeneous deployments", in *ICC 2019 - IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–7.

[2] A. P. Iyer *et al.*, "Automating diagnosis of cellular radio access network problems", in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '17, Snowbird, Utah, USA: ACM, 2017, pp. 79–7.

[3] T. Taleb *et al.*, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration", *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, May 2017.

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

[5] X. Foukas *et al.*, "Orion: RAN slicing for a flexible and cost-effective multi-service mobile network architecture", in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '17, Snowbird, Utah, USA: ACM, 2017, pp. 127–140.

[6] C.-Y. Chang *et al.*, "Radio access network resource slicing for flexible service execution", in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 668–673.

[7] R. Kokku *et al.*, "NVS: A substrate for virtualizing wireless resources in cellular networks", *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1333–1346, Oct. 2012.

[8] M. Andrews *et al.*, "Providing quality of service over a shared wireless link", *IEEE Communications Magazine*, vol. 39, no. 2, pp. 150–154, Feb. 2001.

[9] F. Capozzi et al., "Downlink packet scheduling in LTE cellular networks: Key design issues and a survey", *IEEE Communications Surveys Tutorials*, vol. 15, no. 2, pp. 678–700, Jul. 2013.

[10] N. Nikaein et al., "OpenAirInterface: A flexible platform for 5G research", *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 33–38, Oct. 2014.