Data-Driven RAN Slicing Mechanisms for 5G and Beyond

Sihem Bakri*, Pantelis A. Frangoudis‡, Adlen Ksentini*, and Maha Bouaziz§
*EURECOM, Sophia Antipolis, France
‡Distributed Systems Group, TU Wien, Vienna, Austria
§University of Strasbourg, France
Email: *{name.surname}@eurecom.fr, ‡pantelis.frangoudis@tuwien.ac.at, §maha.bouaziz@unistra.fr

***Abstract*—One of the main challenges when it comes to deploying Network Slices is slicing the Radio Access Network (RAN). Indeed, managing RAN resources and sharing them among network slices is an increasingly difficult task, which needs to be properly designed. The goal is to improve network performance and introduce flexibility and greater utilization of network resources by accurately and dynamically provisioning the activated network slices with the appropriate amounts of resources to meet their diverse requirements. In this paper, we propose a data-driven RAN slicing mechanism based on a resource sharing algorithm running at the *Slice Orchestrator* (SO) level. This algorithm computes the necessary radio resources to be used by each deployed network slice. These resources are adjusted periodically based on current estimates of achievable throughput performance derived from channel quality information, and in particular from the Channel Quality Indicator (CQI) values of the users of each network slice retrieved from the RAN. CQI information is reported to base stations by the User Equipment (UE) following standard procedures, but extracting and frequently reporting it from base stations to the SO may result in significant communication overhead. To mitigate this overhead while maintaining at the SO level an accurate view of UE channel qualities, we propose a machine learning approach to infer the stability of UE channel conditions, as well as predictive schemes to reduce the CQI reporting intensity based on the inferred channel status. Through extensive simulations, we demonstrate the efficiency of our data-driven RAN slicing framework, which allows to meet the stringent requirements of two main classes of network slices in 5G, i.e., enhanced Mobile Broadband (eMBB) and Ultra-Reliable and Low-Latency Communication (URLLC).**

***Index Terms*—Network slicing, radio resource sharing, RAN monitoring, CQI, machine learning.**

I. Introduction

The new generation of mobile networks, known as 5G, is expected to be launched by the end of 2020, promising the support of several novel use-cases coming from other industry sectors (or vertical industries), such as Industry 4.0, autonomous driving, entertainment, etc., in addition to the classical broadband connectivity. Building on the concept of network virtualization, Network Slicing is considered as one of the main enablers of 5G. It allows sharing a common physical infrastructure through building virtual network instances (network slices) tailored to meet specific service requirements.

5G supports three types of network slices [1], namely enhanced Mobile Broadband (eMBB), Ultra-Reliable and Low-Latency Communication (URLLC), and massive IoT (MIoT). Each network slice covers a set of services having the same requirements in terms of Quality of Service (QoS). Generally, a network slice is described and composed by a set of virtual and physical resources, in the form of Network Functions (known as VNF and PNF), deployed and interconnected together on top of a shared infrastructure. Indeed, VNFs include Core Network functions and slice-owner's network services, while a PNF is an already deployed entity, such as a base station (BS) component. The network slice's VNFs and PNF are deployed over different technological domains: Radio Access Network (RAN) and Cloud domain (including the Edge). Mostly, VNFs are deployed on Cloud and Edge domains, while PNFs mainly pertain to the RAN, e.g., gNodeB (gNB) baseband units. All the running network slices are isolated from each other, even though they share the same physical infrastructure.

Management and orchestration of network slices is a critical task, involving making slice life-cycle management decisions at run-time based on monitoring feedback from the service and infrastructure levels. Different slice components at the RAN, transport, and core network need to be coordinated in such a way that target key performance indicators (KPIs) for throughput, latency, availability, and other metrics are attained. The increase in the number of coexisting slices and the number of diverse slice components that need to be monitored already come with significant overhead and the strain on the slice management and control planes is only expected to grow in beyond-5G settings. While the 5G network is maturing, the discussion about how future generations will look like has been kicked off [2], [3]. The general consensus is that the 5G network management architecture needs to evolve to meet complexity and heterogeneity-related coordination challenges by dealing with complex monitoring, analysis, and decision making and becoming more intelligent by natively supporting *AI-driven* operation. This need is a consequence of the expected increase in the dynamics of the network as a result of extreme device mobility, massive densification, and the

potential for large numbers of short-lived personalized slices with diverse requirements. This calls for a self-sustaining network [4] that can autonomously maintain its KPIs under the dynamicity and complexity brought about by a rich set of new beyond-5G applications.

Our work aims to address the above challenges for RAN slice orchestration, particularly focusing on providing intelligent and adaptive RAN slicing mechanisms with reduced monitoring overhead. We build on our prior work [5], [6] to define a data-driven resource management mechanism to support RAN slicing. For this reason, we follow the concept of a two-level MAC scheduler, initially introduced in [7] and [5], that shares the physical radio resources (i.e., Physical Resource Blocks - PRBs) among slices by abstracting PRBs and using two scheduler levels: (i) 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); (ii) the second level, considers the slice-specific (virtual) resource assignment and maps it to actual PRBs, where it controls the number of PRBs assigned to each slice according to the recommendation of a Slice Orchestrator (SO) since the number of PRBs (N_{PRB}) is limited. The second level indicates the maximum number of PRBs to be assigned to each slice, which is derived by the SO. However, the authors do not detail how the N_{PRB} for each slice is computed.

This aspect in particular–namely, *deriving and dynamically adjusting slice resource shares appropriately in response to a changing radio environment*, is the main focus of this work. In order to do so, we rely on monitoring information from the RAN, and in particular on feedback about the slice users' channel quality in the form of Channel Quality Indicators (CQI) obtained from base stations, as we have shown in [6]. This information is directly related to the performance enjoyed by slice UEs and is a critical input in order to calculate the amount of physical resources to allocate per slice to meet its target KPIs. A significant issue with such an approach is the overhead induced by the frequent CQI information exchange between base stations and the SO to tune N_{PRB}. In this work, we tackle these issues and make the following contributions:

- (§III) We provide mechanisms to derive the number of PRBs needed by URLLC and eMBB slices to match their latency and throughput requirements, respectively. For URLLC, we apply tools from queuing theory and devise a way to determine the service rate necessary to attain specific latency targets. These mechanisms require estimates of the achievable data rates of slice UEs, which in turn depend on UE channel quality. For this, we exploit per-user RAN-level information, namely CQI reports collected from base stations [6].
- (§IV) We address the challenge of obtaining an accurate view of channel conditions, which is crucial input for our mechanisms, with reduced monitoring overhead. To this end, we first propose a ML-based mechanism (introduced in [8]) to detect UE channel stability. Our intuition is that for UEs whose channel is not characterized by significant quality fluctuations, savings in terms of monitoring traffic can be achieved by reducing monitoring frequency without significant loss of information.
- (§V) Based on whether a UE is considered to have a stable channel or not, we propose mechanisms to tune the CQI reporting frequency among BSs and the SO, as well as to predict future CQI values without actually retrieving them; for the latter, our approach is based on Long Short-Term Memory (LSTM) neural networks.
- (§VI) We evaluate our mechanisms extensively via simulation, and show them to: (i) detect channel stability with more than 92% accuracy on simulated data, also producing correct classification results on a publicly available LTE measurement data set for different UE mobility scenarios, (ii) outperform static per slice resource sharing in terms of throughput and latency for coexisting eMBB/URLLC slices, (iii) offer significant savings (up to more than 95%) in terms of CQI monitoring overhead for most reporting frequency configurations, at the cost of an average estimation error of less than 7%, and (iv) have low execution time, in the order of milliseconds.

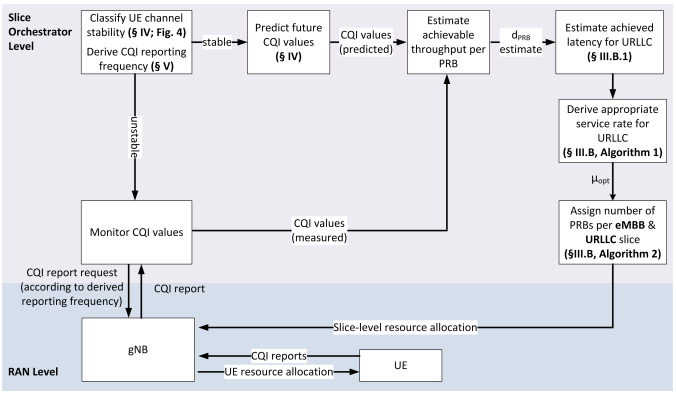


Fig. 1: High-level view of our design and overview of our contributions.

Fig. 1 presents an overview of our work, focusing on the different functional components of our slicing design and how they interact towards providing data-driven RAN slicing.

The rest of this article is structured as follows. Section II reviews the relevant literature. Section III presents our RAN slicing framework, including our channel quality-aware algorithms for slice resource allocation. To reduce the overhead of channel quality reporting, a first step is to be able to classify the status of each UE's channel, for which we provide ML-driven algorithms in Section IV. Then, in Section V, we devise predictive techniques in order to minimize the reporting frequency. The performance of our mechanisms is evaluated in Section VI before we conclude the article in Section VII.

II. Related work

Resource allocation among slices has become a challenge that requires efficient mechanisms allowing sharing resources between slices in a (near-)optimal way, according to their

heterogeneous needs and to the limited bandwidth. Several works in the literature address issues in this space.

Vo et al. [9] focus on RAN slicing and slice coordination by formulating a bi-convex problem, which accounts for dependencies between radio, base station caching, and backhaul resource allocation. They present two alternative minimization algorithms to solve the proposed bi-convex problem. Even though their alternative minimization approaches have no global convergence guarantees, their simulation results show that they can achieve a global optimal solution. Unlike our work, the proposed mechanism does not consider slice latency requirements at the RAN and thus cannot directly support URLLC. Moreover, to derive the ratio of radio resources to allocate per slice, information on each user's downlink rate is necessary by the slicing algorithm, which is calculated from each user's SNR. It is this type of channel quality-related feedback that our mechanisms aim to minimize.

Zhang et al. [10] present a resource allocation mechanism for a two-tier (macro-cell vs. small cell) sliced system, adapted to the QoS requirements and interference constraints of URLLC, eMBB, and IoT slices. They address the problem of allocating resources to maximize the *uplink* capacity on each sub-channel for small cells, taking into account constraints on the transmit power of each cell user, the data rate required by the (URLLC) users, and the interference among cells. The authors transform the problem to a relaxed, continuous convex version, which they solve using the Lagrangian dual decomposition method. Apart from the different problem settings and architectural assumptions, their mechanisms operate at the sub-channel and transmission interval levels. In our design, these allocation decisions map to the lower scheduler level. Our mechanisms, on the other hand, aim at estimating and dedicating the physical resources necessary per slice in a channel quality and latency-aware manner, which is part of the input for the aforementioned decisions.

Arand et al. [11] study joint eMBB and URLLC schedulers by designing optimization algorithms for common scheduling between the traffic of the two considered traffic types, where the objectives are dual: maximizing utility for eMBB traffic while satisfying instantaneous URLLC requests. This is achieved by dynamically multiplexing the URLLC traffic through puncturing/superposition through/over the eMBB traffic. The theoretical and simulation results show that the structural properties of the joint optimization problem allow for clean decomposition, which validates the characteristics and benefits of their proposed schedulers. Since our decisions take place at the slice orchestrator level, our mechanisms are to an extent complementary and could be considered jointly. At the orchestrator level, we decide on a (minimum) amount of resources per slice, and then the scheduler at the gNB can multiplex them either via puncturing or superposition, as Arand et al. describe, or by dedicating an amount of resources per slice according to the decision of the slice orchestrator and having two independent slice-dedicated schedulers. These mechanisms pertain to resource partitioning within a slice. In this article, we do not use channel state information for (intra-slice) per user resource allocation and transmission scheduling decisions, but, rather, for *per slice* resource allocation.

In a non-cooperative network slicing setting, Caballero et al. [12] adopt a game-theoretic approach, and propose and analyze dynamic resource sharing mechanisms, which allow slices to unilaterally customize resource allocations to their users. The proposed network slicing framework combines admission control with resource allocation and user dropping within a slice so that each slice maximizes its utility subject to its share of network resources. Our work assumes a different slicing framework, where it is the task of the slice orchestrator to dynamically tune slice resource shares based on RAN monitoring data. Distributing the resource share of a slice to its individual users, a task of the slice-specific scheduler level of our design, as well as admission control, are not in the scope of this article.

Sciancalepore et al. [13] present slice resource allocation algorithms consisting in: i) prediction and traffic analysis per network slice using the Holt-Winters forecasting procedure, to analyze and predict future traffic requests associated with a particular network slice, ii) a heuristic scheme in order to make admission control decisions for network slice requests, and iii) multi-class traffic scheduling and adaptive correction of the previous forecasting solution according to measured deviations. Their work is similar in spirit to ours, in the sense that they apply forecasting schemes for slice-level resource management. However, while they aim at forecasting slice traffic demands to perform admission control and resource allocation, we address a different problem: We assume fixed traffic requirements at the application level and apply prediction mechanisms to estimate the necessary slice resources in a channel-quality aware manner. Furthermore, their algorithms are executed less frequently, i.e., in the order of hours (at slice admission), and thus their running time, which can reach hundreds of seconds, does not pose a problem. Our mechanisms are designed to be executed at much finer timescales and have lower run-time complexity.

Finally, Salvat et al. [14] propose a joint slice admission control and resource reservation framework, including a hierarchical control plane for end-to-end slice orchestration and adopting revenue management models already used in other contexts, such as the airline industry. In particular, they introduce the concept of slice resource overbooking to maximize the revenues of mobile operators and proposed exact and heuristic algorithms to solve the respective optimization problems. Their approach also makes use of runtime monitoring information from the RAN to map signal quality to achievable rates and to perform predictions (for the latter, an approach similar to [13] is used). However, aspects related to optimizing this monitoring frequency, which are central to our work, are not discussed.

In conclusion, the state-of-the-art techniques and mechanisms proposed for dynamic slice resource sharing are based on the formulation and solution of optimization problems considering individual or combined objectives related to several constraints imposed by users, operators, etc. These problems

are typically NP-hard, and thus, solving them in real-time brings up significant practical challenges. The mechanisms we present in this article have low complexity, are based on simple channel status estimation models that rely only on RAN-level data directly available to the mobile network operator and are thus suitable for run-time operation. Our system is supported by predictive mechanisms, which are shown to precisely estimate the amount of resources needed by each slice while reducing the associated monitoring overhead. Notably, reducing the monitoring intensity for RAN slicing is an understudied topic in the related literature.

III. RAN Slicing framework

A. Architecture and assumptions

In this work, we assume a 5G network that includes a set of deployed base stations (gNBs, in 5G terminology) covering an area, and a Slice Orchestrator (SO) that is responsible for deploying and managing the life cycle of network slices in the mobile network (RAN and core network). Thus, we consider the same type of network architecture adopted in [7] and [5]. Besides, we assume that a SO is responsible for a region covered by a set of gNBs. The communication between the SO and the gNBs is realized using a southbound communication protocol, such as FlexRAN [15], which allows to interact and remotely manage the gNBs. The gNB management process consists of obtaining information on the status of the RAN and configuring the gNBs in an appropriate way, for example, by determining the number of PRBs allocated to each slice. Then, we assume that a set of UEs is served by (or associated with) a network slice spanning a set of gNBs (i.e., different physical locations). The SO receives a request from a tenant (slice owner) to instantiate a slice in the form of a slice template, which may indicate the slice duration, its type (e.g., eMBB, URLLC, MIoT), the list of involved UEs, the application requirements (for example the maximum tolerated latency) and the application data rate (denoted by λ) of the service used in this slice. Based on this information, the SO determines the appropriate number of PRBs that meets the requirements of the slice, which will be communicated later to the concerned base stations via the southbound control protocol (e.g., FlexRAN).

We focus on network slices that belong to either the URLLC or the eMBB service class and propose two corresponding mechanisms to estimate the required N_{PRB} for each slice type. The principal difference between URLLC and eMBB is that URLLC requires low latency, while the eMBB is aimed at high data throughput. For MIoT, we assume semi-persistent scheduling as indicated in [5].

B. Proposed algorithms

Since URLLC and eMBB pose different performance requirements (guaranteed latency vs. data rate) for each slice type, we propose a different mechanism to estimate the amount of radio resources necessary to meet these requirements. In both cases, we begin with an initial estimate of the required number of PRBs (N_{PRB}) solely based on information that is provided at instantiation time by the slice tenant in the slice template (e.g., UEs in a slice, application bitrate). This takes place under an optimistic assumption about each UE's rate capabilities and in a manner unaware of their actual channel conditions, which we initially assume to be ideal. However, it is these conditions, combined with the amount of radio resources allotted to each UE, that actually determine the achievable latency and throughput performance. Therefore, we provide algorithms that continue by dynamically improving this estimate, periodically tuning N_{PRB} based on the CQI feedback obtained from gNBs.

1) URLLC slice: The proposed algorithm aims to derive the necessary service rate per gNB to maintain the latency requirements of a URLLC slice (Lat_{max}) and translates it into an initial PRB allocation. This mechanism relies only on information obtained from a slice template provided by the slice tenant and is agnostic to the channel conditions of each UE.

Since each slice has its own downlink queue at the gNB [5], all packets belonging to the slice share the same queue. Therefore, we propose to model the slice queue at the gNB as an M/M/1/K [16, Chapter 3.6] one in order to estimate the latency of the packets. We should note that the M/M/1/K model has been recently applied by Kozat et al. [17] in the context of slice availability, while Castagno et al. [18] evaluate it as one of various candidate models to approximate the throughput and blocking probability in general cellular radio systems.

In our model, the service rate (μ) is exponential, the traffic arrival rate follows a Poisson distribution with intensity λ, and the queue has a size of K. Here, the service rate μ depends on the scheduling process at the MAC layer, while the value of λ corresponds to the traffic rate of the application running on top of the slice. To derive μ and λ, we use the following formulas:

$$\lambda = \frac{N_{users} d_{App/user}}{L} \tag{1}$$

$$\mu = \frac{N_{PRB} d_{PRB}}{L} \tag{2}$$

where:

- N_{users} is the number of UEs belonging to the slice and connected to the gNB.
- L denotes the average packet size of the URLLC application.
- $d_{App/user}$ is the data rate per user required by the application running on top of the URLLC slice, considering the same value for all slice users.
- d_{PRB} is the maximum data rate provided by one PRB.

We apply Little's law to estimate the latency experienced by URLLC packets. This law assumes that the average time a user spends in a queue depends on the number of active users and the traffic intensity, whatever the distribution of the arrival rate. Since, in our case, the number of users corresponds 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:

$$T_w = \frac{N_{packet}}{\lambda} \tag{3}$$

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

$$N_{packet} = \frac{1-\rho}{1-\rho^{K+1}} \sum_{k=0}^{K} k\rho^k \tag{4}$$

where, $\rho = \frac{\lambda}{\mu}$.

Since μ depends on the number of resources dedicated to the URLLC slice and corresponds to the service rate of the URLLC queue, 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 Lat_{max} which is assumed to be the maximum tolerated latency by a URLLC slice:

$$T_w \leq Lat_{max}. \tag{5}$$

We substitute T_w by its value given by (3), obtaining the following expression:

$$\frac{N_{packet}}{\lambda} = \frac{\frac{1-\frac{\lambda}{\mu}}{1-(\frac{\lambda}{\mu})^{K+1}} \sum_{k=0}^{K} k(\frac{\lambda}{\mu})^k}{\lambda} \leq Lat_{max} \tag{6}$$

Therefore, we need to find a value of μ, noted μ_{opt}, that ensures condition (5). According to (2), we can extract the number of $PRBs$ (noted N_{PRBopt}) to dedicate to a URLLC slice as follows:

$$N_{PRBopt} = \frac{\mu_{opt}\ L}{d_{PRB}}. \tag{7}$$

At this step, to derive μ_{opt} we aim to solve (6) for μ, by taking into account the assumption that L is constant and the value of d_{PRB} is the same for all UEs. Deriving μ_{opt} analytically is hard; hence, we numerically estimate it using the following simple algorithm.[1]

The steps of Algorithm 1 are detailed as follows: First, we generate n candidate values for μ and keep them in a vector Mu. These values for μ can be generated for each possible number of PRBs defined by the available bandwidth for the given radio technology using (2). Then, we check if condition (5) is respected, by calculating N_{packet} corresponding to each value of μ and then comparing the resulting value T_w with Lat_{max}.

Note that during the comparison of T_w with Lat_{max} to accept or reject a μ value, we use a latency margin ϵ to ensure that T_w is adequately lower than the latency threshold Lat_{max}, and also close enough to it in order to respect condition (5) without wasting a lot of radio resources.

Among all the μ values that lead to an acceptable latency, we select as the optimal the one which minimizes the difference between Lat_{max} and T_w, i.e., the smallest value of M_{opt}.

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

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

Finally, we use (7) to derive the corresponding N_{PRB} to be assigned to a URLLC slice, once μ_{opt} is obtained.

2) eMBB slice: The main constraint of estimating N_{PRB} is that an eMBB slice requires a high data rate. To satisfy it, the number of PRBs N_{PRBmax} to dedicate for an eMBB slice periodically at each gNB should be equal to (or greater than) the aggregate data rate needed by the slice application for all slice UEs connected to it. This is captured in (8).

$$d_{PRB}\, N_{PRBmax} = d_{App/user} N_{users}, \tag{8}$$

where,

- N_{users} is the number of UEs of the eMBB slice connected to the gNB.
- $d_{App/user}$ is the data rate per user required by the application running on top of the eMBB slice.

This equation indicates that the N_{PRBmax} allowed to a slice on a given gNB should cover the required slice's applications (i.e., the data rate required by the application multiplied by the number of active users N_{users}).

We assume that $d_{App/user}$ is the same for all users, d_{PRB} is the maximum data rate provided by one PRB, and that it is the same for all UEs. In addition, we consider PRB rate as 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. Once each N_{PRBmax} value is calculated using (8), it is communicated by the SO to the corresponding gNBs via the southbound protocol.

3) A channel quality-driven algorithm for dynamic N_{PRB} estimation: In the previous step, the calculation of the initial number of resource blocks per slice (URLLC and eMBB) is based on the assumption that d_{PRB} is fixed for all users. However, users have different channel conditions and, thus, different data rates. In order to correct the estimation of the d_{PRB}, we propose to use information from per-UE channel

quality reports obtained from gNBs. In 4G and 5G, the CQI reports are transmitted from UEs to eNBs and gNBs, respectively, via a standard procedure in order to be used in the scheduling process, and they include the CQI and MCS values of each UE belonging to a cell. This information should be made available to the SO and be used as input to our algorithms. While there is no standardized procedure to do so, solutions tailored to specific base station implementations exist, such as FlexRAN, which is integrated into the OpenAir-Interface[2] open-source platform and has been the southbound protocol we have used in prior work [19].

Based on the CQI, we can estimate d_{PRB} per UE and per cell (gNB). Indeed, d_{PRB} can be obtained based on the same tables used by the gNB to translate a CQI to a data rate [20] (translation table). Therefore, we organize these CQI values in a matrix $v(j,k)$, where k is the id of the UE and j is the id of the slice. Then, using the translation table, we transform the matrix v to a matrix of data rates noted $d_{PRB}(j,k)$, where j and k have the same meaning as for matrix v.

Afterward, a dynamic slicing algorithm is introduced, for both eMBB and URLLC slices, to exploit the RAN-level information per user to more accurately translate the service rate derived into an appropriate PRB allocation.

The different steps of the dynamic slice resource allocation procedure are presented in Algorithm 2. We note that $N_{PRBopt}(i,j)$ is a matrix that gives for each cell i the necessary number of PRBs for slice j, $Slice(j)$ gives the type of the deployed slice, $d_{App_user}(j)$ is the data rate required by an application (per user) running on top of a slice j, and $N_{users}(i,j)$ is a vector indicating the number of users of a slice j in cell i. This algorithm allows a more accurate estimation of the N_{PRB} allocated to each slice and for each network cell: For URLLC, it applies equation (7), using the optimal service rate as determined by Algorithm 1 to meet the latency requirements, and the mean achievable d_{PRB} across all slice users per gNB considering each user's channel quality, instead of a fixed optimistic value for all. For an eMBB slice, it sums up the resources required per UE by considering the individual radio capacity of each user reflected in $d_{PRB}(j,k)$.

Note that this algorithm is run by the SO periodically, and it relies on periodically transmitted CQI reports from gNBs. The periodicity of these algorithms is independent of the scheduling period TTI used at the MAC layer of gNBs.

C. Limitations of the proposed solution

The previously proposed algorithm is based on the periodic CQI reports sent by gNBs to the SO (refer to Fig. 1), in order to correctly update d_{PRB}.

However, this CQI reporting may involve significant traffic overhead and can become an issue with a high number of slices or UEs. Hence, optimizing this signaling process is crucial and represents a challenge that we are addressing in the next sections.

[2]https://www.openairinterface.org/

Result: $N_{PRBopt}(i,j)$
for *each cell* i **do**
  for *each slice* j **do**
    if Slice (j) == eMBB **then**
      $N_{PRBopt}(i,j) = \sum_{k=1}^{N_{users}(i,j)} \frac{d_{App/user}(j,k)}{d_{PRB}(j,k)}$
    else
      if Slice (j) == URLLC **then**
        $N_{PRBopt}(i,j) = \frac{\mu_{opt} L}{\frac{1}{N_{users}(i,j)} \sum_{k=1}^{N_{users}(i,j)} d_{PRB}(j,k)}$
      end if
    end if
  end
end

Algorithm 2: Calculation of N_{PRB} for eMBB and URLLC slices for multiple cells

IV. Channel stability prediction using machine learning

A. Problem and objectives

As indicated earlier, the main issue of the precedent algorithm is the potential traffic overload due to frequent signaling messages (CQI reports between gNBs and the SO). Therefore, a key challenge consists in proposing mechanisms to optimize this procedure, reducing the frequency of CQI report transmissions.

The idea focuses on limiting the amount of CQI reports and avoiding unnecessarily transmitted ones while ensuring that the SO maintains an accurate view of the state of the channel. To this end, we apply ML mechanisms in order to predict channel stability/mobility, as this can be used to decide if a CQI value is necessary to be reported, and in turn, to control the reporting frequency. For instance, if the channel is stable, it is not necessary to frequently retrieve the CQI report since it does not vary during this time period. In this context, we introduce two algorithms: the first one allows to estimate the appropriate frequency of reports while minimizing the d_{PRB} estimation error as much as possible, i.e., by reducing the CQI collection frequency; the second is based on a ML-based prediction method, namely Long Short-Term Memory (LSTM), and has the purpose of forecasting a sequence of CQI or d_{PRB} values during a time interval based on past CQI or d_{PRB} values respectively, without the need to retrieve the actual ones from the gNB.

B. Channel stability prediction

1) Overview: In order to provide the information about the channel quality, CQI messages of each slice UE are sent periodically from the gNB to the SO, allowing to appropriately drive the number of resource blocks N_{PRB} for each slice. However, when the channel quality is relatively stable, the CQI values do not vary much in time. Therefore, frequent CQI reports will not contribute to improving the view of the

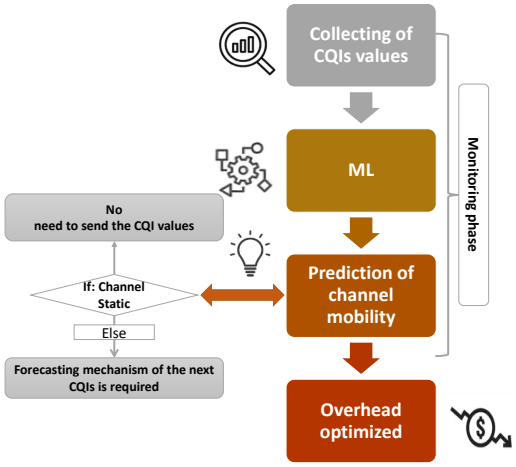


Fig. 2: Concept and methodology to reduce CQI monitoring overhead.

SO on the actual radio conditions of UEs, and hence it does not affect the quality of the per-slice radio resource allocation.

Our approach consists of monitoring the channel state for a period of duration of T. If the channel is identified as "mobile" by the predictor, a forecasting mechanism of the next CQI values will be needed, or a new CQI value is required to be retrieved from the gNB in order to adjust the resource allocation. Otherwise (i.e., the channel is identified as stable), there is no need to receive (or forecast) new CQI values. Thus, the SO considers the last received CQI value as accurate to allocate radio resources, which reduces the CQI reporting frequency. The different steps of this concept are illustrated in Fig. 2.

The proposed monitoring phase is based on a ML algorithm, which helps to classify the channel state as mobile or stable. In [8], after testing some ML algorithms, we selected Support Vector Machines (SVM) and Neural Networks (NN), as they were found to offer better accuracy.

2) Machine learning architecture: This phase consists of collecting data and then processing them to extract specific features and create feature vectors (also called characteristic vectors), which will be used to train a classifier. Raw data is collected into vectors for different channels during a period τ of duration T. Then, for each data vector (i.e., for each channel), a feature vector is created. In order to represent the state of the channel using ML algorithms, several types of data (such as SNIR, CQI, and others) can be used. However, the collection of different types of data may not always be available or easy due to several constraints, such as confidentiality, security, financial, and others. To this end, and to avoid using several data types, we use only a single type (the CQI value). We collect CQI values, and then, in order to extract the appropriate feature vector, we perform a pre-processing step on this vector to obtain the relevant data for the predictive system to be used, to identify the state of the channel (mobile or stable).

Pre-processing is performed on the data vector $CQI_\tau = [cqi_1, cqi_2 ... cqi_n]$ of n CQI values gathered during period τ in order to extract the characteristic vector C= [C_1 C_2 C_3]. The extracted features (or characteristics) are as follows:

- C_1: The difference between the maximum and minimum values of collected CQIs in the data vector CQI_τ.

$$C_1 = cqi_{max} - cqi_{min} \tag{9}$$

  If C_1 is small or zero, the channel may be static, which means that the environment is stable because there are no significant effects causing a drastic change in the CQI value. This feature can give an idea of the channel state.
- C_2: Variance.

$$C_2 = \frac{1}{n}\sum_{i=1}^{n}(cqi_i - \overline{CQI_\tau}) \tag{10}$$

  This feature measures the dispersion of CQI values relatively to their average $\overline{CQI_\tau}$.
- C_3: The vertical change of the CQI curve slope, representing the CQI change in different samples in period τ.

$$C_3 = \mid CQI(t_{i+\Delta}) - CQI(t_i) \mid \tag{11}$$

  where, $CQI(t_i)$ and $CQI(t_{i+\Delta})$ are the CQIs collected at time instances t_i and $t_{i+\Delta}$ respectively, and Δ=5 in our case. Multiple C_3 values are extracted for each sample. Thus, the size of C depends on the number of C_3 values.

It is worth noting that, our feature modeling approach follows common practice in research areas such as pattern, image and voice recognition, where recognition is most often based on statistical features such as the Mean Absolute Value (MAV), Zero Crossing (ZC), Slope Sign Changes (SSC), and Root Mean Square (RMS) [21], [22], [23].

After the creation of vector C, a known label (stable or mobile) is assigned to it in order to be used for the training phase.

In this ML system, we have used 70% of the feature vectors with their labels to train the classifier; the rest of the feature vectors (30%) are used during the test and validation phase. Fig. 3 presents the different steps involved in the channel state prediction process.

As mentioned above, this method aims at predicting the channel mobility over an interval τ. The next step is to estimate the frequency of the CQI reports over the following interval $\tau + 1$, for which we propose in the next section two methods to reduce the frequency of CQI reports while minimizing errors between real and predicted d_{PRB}.

V. Algorithms for Reducing the CQI Reporting Frequency

This section presents two methods to reduce the number of CQI report exchanges, noted N_r, and expressed as follows:

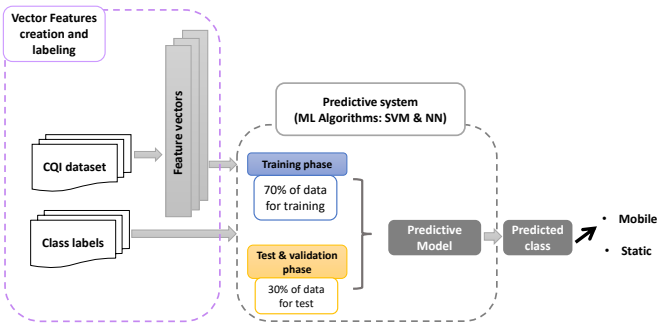


Fig. 3: ML-driven channel stability prediction.

$$N_r = \frac{T}{f_{rep}}, \tag{12}$$

where T is the duration of the test time period, and f_{rep} is the time interval between two successive CQI reports. This means that reducing N_r is equivalent to increasing f_{rep}.

Our objective here consists in reducing N_r (or increasing f_{rep}) of CQI report message exchanges as much as possible, while reducing the error between the real d_{PRB} (using CQI reports) and the predicted one (when CQI is predicted).

Our system alternates between *monitoring* and *prediction* periods. For this purpose, we consider a (fixed-duration) time interval (τ) when a monitoring phase takes place by collecting a fixed number of CQI samples to evaluate the stability of the channel. On the one hand, if the channel is stable, there is no need to recover the CQI report for the next interval $\tau + 1$, which helps reduce the frequency of the CQI report exchange. On the other hand, if the channel is classified as mobile, we apply one of the proposed methods to either (i) predict the optimal number of CQI reports required for the next interval of equal duration or (ii) forecast a sequence of CQI values without actually retrieving them. These two methods are called *Optimal Difference* and *Long Short-Term Memory (LSTM)*, respectively, and are described in the following.

It is worth noting that for LSTM the duration of each prediction period depends on the system QoS required. For example, the operator may decide to shorten a prediction period to sacrifice monitoring load gains in order to achieve a more accurate view of the actual UE channel conditions.

A. Optimal Difference method

This method is based on estimating the stability of the channel over a period τ and then selecting the appropriate number of CQI report exchanges N_r for the next period $\tau+1$, while minimizing the error (E) between real and estimated d_{PRB} corresponding to actual and predicted CQI values.

The challenge here consists in minimizing N_r and E, which cannot be solved by an optimization algorithm, as we do not have exact constraints on the error. Hence, we need to find a relation between E and N_r that allows to define the optimal N_r and E. Obviously, if N_r decreases (f_{rep} increases), the error E either remains the same or increases. Indeed, when N_r decreases, the frequency update of the d_{PRB} values decreases and consequently causes errors mainly for high-mobility channel cases.

For this reason, we first generate the vector N_r which consists of a set of m values for N_r, each representing a different CQI report exchange rate. The first N_r value in the vector corresponds to a value noted $N_r^{(1)} = n$, which is the maximum number of CQI samples that may be collected during the prediction period.

To generate the rest of the values, we use the geometric progression method with ratio q as follows:

$$N_r^{(i+1)} = qN_r^{(i)} \tag{13}$$

In order to obtain decreasing N_r values, we assume that $0 < q < 1$ and select $q = 1/2$, which allows to observe the impact of decreasing N_r on E. Next, after creating the vector $N_r = (N_r^{(1)}, N_r^{(2)}, \ldots, N_r^{(m)})$, we can deduce the corresponding error of each N_r value and consequently deduce the error vector $E = (E_1, E_2, \ldots, E_n)$.

We proceed as follows: At the end of monitoring period τ, we have collected vector CQI_τ, which we translate to the corresponding sequence of d_{PRB} values noted as $P_\tau = [p_1, p_2, \ldots, p_n]$. Then, out of this sequence of actual d_{PRB} values, for each $N_r^{(i)}$ we generate a sequence $\hat{P}_\tau^{(i)} = [\hat{p}_1^{(i)}, \hat{p}_2^{(i)}, \ldots, \hat{p}_n^{(i)}]$ of *estimated* ones by keeping only every $i-$th value from P_τ, and replacing the rest with their preceding real d_{PRB} value. For example, for $i = 2$ we have that $N_r^{(2)} = \frac{N_r^{(1)}}{2}$, thus $\hat{P}_\tau^{(2)}$ will be composed by keeping every second actual value from P_τ and setting every other value to its precedent, i.e., $\hat{P}_\tau^{(i)} = [p_1, p_1, p_3, p_3, p_5, p_5, \ldots]$. Finally, the error E_i is given by:

$$E_i = \frac{1}{n}\sum_{j=1}^{n} |p_j - \hat{p}_j^{(i)}|. \tag{14}$$

The idea behind the creation of N_r and E is to normalize them by their maximum, and to calculate the difference between their normalized values $\Delta_i = |\ \tilde{N}_r^{(i)} - \tilde{E}_i\ |$. Then, we select the N_r that corresponds to the minimum difference Δ_{min} between E and N_r, allowing to estimate the optimal N_r (N_r^{opt}) and consequently the optimal error. The steps of this method are illustrated in Figure 4.

B. Long Short-Term Memory method

Our second approach to reduce the reporting frequency is based on using the Long Short-Term Memory (LSTM) method to forecast d_{PRB} values for period $\tau+1$, i.e., *without* actually collecting any CQI values during $\tau + 1$.

LSTM is a deep learning method (a type of Recurrent Neural Network (RNN)), which can learn the long term dependencies between time steps in time series, and sequence data. Its purpose consists in predicting the values of the future time steps of a sequence. LSTM has good performance in predicting long-interval events and processing long-term dependencies

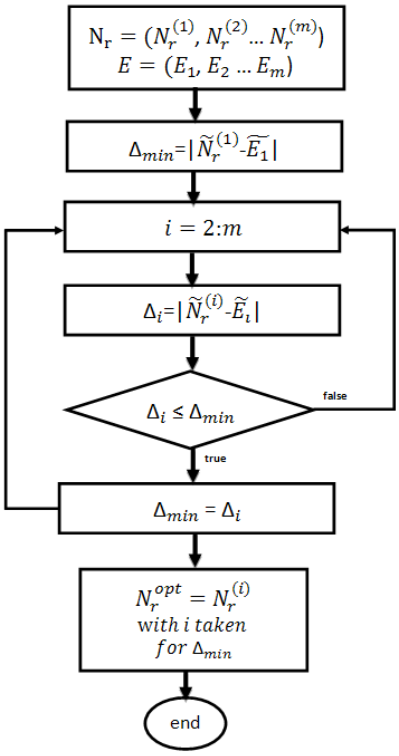


Fig. 4: Steps to estimate the optimal number of CQI reports N_r^{opt}.

of time series data [24]. Besides, it presents advantages in terms of classification accuracy, where it outperforms several traditional time series classification models.

In our case, the data sequence is constituted by the values of CQI or d_{PRB}, where we use LSTM to predict CQI or d_{PRB} values for period $\tau+1$ based on the respective values collected during period τ.

C. Comparison between methods

After a monitoring period τ where a sequence of CQI reports is collected and the channel status is characterized, one of the two alternative strategies that we propose to reduce reporting frequency may be applied. The Optimal Difference method consists in estimating the appropriate number of CQI reports (N_r) to be collected during the next prediction period $\tau + 1$ which we consider here to have a fixed duration equal to that of τ, while the LSTM method consists in predicting a sequence of d_{PRB} values for $\tau + 1$ whose duration may vary according to the operator preferences, based only on the d_{PRB} values corresponding to the CQI reports collected during τ. Given the qualitative difference of the two methods, here we introduce appropriate metrics to compare them. First, we focus on the performance of each method in terms of CQI collection-related traffic savings. For this, we introduce two metrics representing the (normalized) CQI reporting rate gain for both methods following equations (15) and (16).

$$G_{OptDif} = \frac{CQIrate_{default} - CQIrate_{Opt}}{CQIrate_{default}} \quad (15)$$

$$G_{LSTM} = \frac{CQIrate_{Pred}}{CQIrate_{default}} \quad (16)$$

- $CQIrate_{default}$ refers to the number of CQI reports during period $\tau + 1$ with duration T without any optimizations (representing the maximum N_r).
- $CQIrate_{Opt}$ refers to the optimized number of CQI reports N_r in the same period.
- $CQIrate_{Pred}$ refers to the number of CQI values predicted during the LSTM forecasting period.

Second, we characterize the d_{PRB} estimation error for each strategy. In particular, for each case, the estimation error E during period $\tau+1$ is defined as the mean (absolute) difference between a d_{PRB} value estimated and the actual one (if the corresponding CQI value were actually retrieved). We further normalize this error as follows:

$$E_{Norm} = \frac{E}{d_{PRBmax} - d_{PRBmin}}, \quad (17)$$

where $(d_{PRBmax} - d_{PRBmin})$ represents the difference between the maximum and minimum d_{PRB}, in order to express the maximum error that a method can detect in its prediction.

The purpose of these metrics is to capture the trade-off between gains in terms of CQI monitoring traffic reduction and d_{PRB} estimation accuracy, and help to identify via our quantitative evaluation the conditions under which each method is more appropriate.

VI. Performance Evaluation

This section focuses on evaluating the performance of the different methods and techniques proposed in this paper. First, we will evaluate the performance of the proposed algorithms for reducing the frequency of CQI reporting presented in Section V, while keeping the d_{PRB} estimation error low. Then, we will assess the efficiency of the data-driven RAN slicing algorithm introduced in Section III, followed by a study on the impact of our mechanisms for reducing the reporting frequency on meeting RAN slice performance requirements.

A. CQI prediction and reporting frequency optimization

1) Channel mobility/stability prediction: In [8], we evaluated the performance of the channel stability predictive system using two ML algorithms (NN and SVM) based on a data set we generated from simulations using `ns-3`. (For a validation of our models on a publicly available data set with measurements from an LTE network testbed, see Section VI-A2.) In this context, we simulated an LTE cell, considering UEs moving with different constant velocities and different distances from the eNB, in order to create a data set with realistic CQI values corresponding to different degrees of user mobility. We generated approximately 15500 vectors of CQI values, and each vector contains around 10000 CQI values for different channel mobility states. Therefore, we

extracted a feature vector for each CQI vector as described in Section IV-B, which we labelled either as static or mobile, depending on the level of channel mobility.

We used MATLAB to train and validate ML algorithms based on the provided CQI data set. Note that, the considered NN algorithm has a single hidden layer. For both ML algorithms, we used 70% of our data for training and the remaining 30% for test and validation. It is worth noting that the training processing time (i.e., training phase duration) of the SVM and NN algorithm considered are respectively 1.15 s, and 2.2620 s. However, the test phase duration is much shorter, in the order of milliseconds.

Table I presents the results of the validation phase in terms of accuracy and F1-score for the two candidate ML mechanisms.

TABLE I: Accuracy and F1-score of NN and SVM algorithm

	NN	**SVM**
Accuracy	96.43%	92.86%
F1-score	96.29%	92.30%

As shown in Table I, both algorithms (i.e., SVM and NN) are able to learn and classify the channel state with high performance. They guarantee an accuracy and F1-score of more than 90%. However, we note that the NN system outperforms the SVM in terms of accuracy and F1-score by only about 4%. Note that accuracy and F1-score are defined as follows:

- Accuracy, i.e., the ratio of the number of correctly predicted vectors to the total number of vectors.

$$\text{Accuracy} = \frac{\text{\# correctly predicted}}{\text{\# feature vectors}} \tag{18}$$

- F1-score, which is defined by the weighted average of *precision* and *recall*, where, *precision* is the ratio of the number of correctly predicted mobile class instances to the total number of predicted mobile class ones (i.e., false and correct), and *recall*, also called sensitivity, is the ratio of the number of correctly predicted instances of the mobile class to the number of all true mobile class ones.

$$\text{F1.score} = \frac{2 * (\text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})} \tag{19}$$

The channel stability/mobility prediction results, shown in Table I, were tested and evaluated using a dataset generated via `ns-3` simulations. We next validate our channel stability prediction mechanisms on real traces from an operating mobile network testbed.

2) Model validation with real data: In this test, we used a data set publicly available from CRAWDAD [25], which contains statistics and monitoring data of 4G/5G MAC, RRC and PDCP layers.

The considered data are raw and recorded for one eNB and a single mobile UE in five different mobility scenarios by following different motion and distance patterns relative to the eNB. All raw data have been recorded without including Tx power amplification on the RF front end (0 dBm transmit power), which implies an approximately 10 m maximum range of coverage. From these data, we have extracted the CQI measurements for each one of the following mobility scenarios:

- Moving Away (MA): the UE moves away from the eNB to a maximum distance of 10 m.
- Moving Closer-Far-Closer (MC): the UE moves back and forth relative to the eNB, from a 0 distance up to approximately 10 m.
- Stable Long Distance (SLD): the UE stands still in a long distance (approximately 5-10 m) away from the eNB.
- Stable Mid Distance (SMD): the UE stands still in a mid distance (approximately 1-5 m) away from the eNB.
- Stable Short Distance (SSD): the UE stands still in a short distance (approximately 0-1 m) away from the eNB.

Then, we use CQI data that correspond to actual UE mobility scenarios to validate the channel stability results deduced previously using the `ns-3` dataset.

The average duration of the test phase of all the considered scenarios using SVM and NN algorithms are: (i) 5.7 ms using SVM and (ii) 210.4 ms using NN. We conclude that our algorithms are fast, and their execution time is acceptable in this kind of analysis. However, when response time matters most, it would be better to use the SVM algorithm.

Table II shows the confusion matrix results in a simplified way,[3] which indicates for each mobility scenario (MA, MC, SLD, SMD, and SSD), the results of the classification, i.e., mobile or static, obtained using NN and SVM.

TABLE II: Mobility and stability results of the considered mobility scenarios using NN and SVM.

	NN	**SVM**
MA	*Static*	*Mobile*
MC	*Mobile*	*Mobile*
SLD	*Mobile*	*Mobile*
SMD	*Mobile*	*Mobile*
SSD	*Static*	*Static*

The obtained results indicate that both SVM and NN have correctly classified the channel of the MC scenarios as mobile. We argue this by the fact that the UE, in this case, is moving closer and away from the eNB, so the CQI values change with it. Also, for the SSD scenario, the channel was well classified as static since the UE remained static and close to the eNB.

For the SLD and SMD scenarios, both classification algorithms classified the channel as mobile, although the UE remained static. We explain this by the fact that for SLD and SMD scenarios, the distance between UE and eNB is long and medium respectively, which strongly affects the channel,

[3]A confusion matrix indicates the number of true and false classifications across the whole validation dataset. As we have only 5 mobility scenarios, we have directly given the classification results obtained at the output of the ML algorithms used for each scenario.

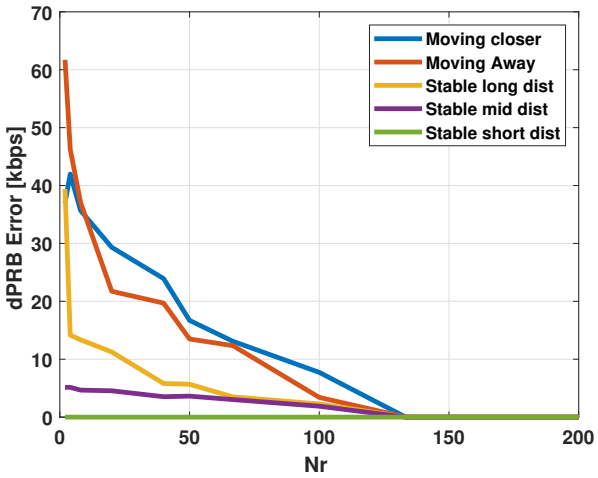


Fig. 5: d_{PRB} error vs. N_r.

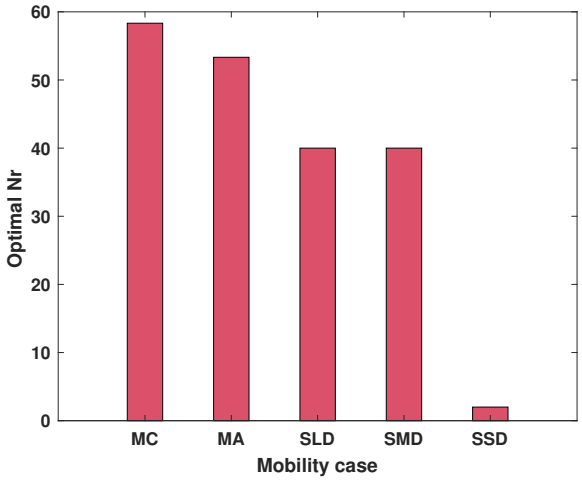


Fig. 6: Optimal number of CQI reports (N_r^{opt}) over periods of $\tau = 200ms$ vs. mobility scenario.

for instance due to multi-path effects, leading to significant variations in the channel quality. Regarding the MA scenario, the SVM correctly classified the channel as mobile, but the NN considered it as static. We conclude that both SVM and NN algorithms perform well, as they have well detected the channel status of the majority of the considered scenarios. However, regarding the response time constraint, it would be better to use SVM because it is much faster than NN.

3) Reducing the CQI reporting frequency:

a) Optimal Difference method results: As elaborated in Section V-A, the Optimal Difference method is a technique that calculates the estimation error of d_{PRB} for different CQI reporting frequencies, i.e., different numbers N_r of messages exchanged and, correspondingly, different inter-report times f_r, over time intervals of length τ. Then, it selects the N_r value that minimizes the difference between the (normalized) values of N_r and the PRB estimation error. In the time interval that follows, the optimal CQI reporting frequency, calculated as described above, is applied. An increase in N_r is equivalent to a decrease in f_{rep} according to Eq. (12).

Fig. 5 shows the errors of the d_{PRB} values obtained for different N_r, in an interval of $\tau = 200ms$. The curves show that when N_r increases, the error decreases. They also demonstrate that the d_{PRB} estimation error for a UE in the MC scenario is the highest compared to the other scenarios for most CQI reporting frequencies. This error is higher than the MA scenario, followed by SLD, SMD, and SSD scenarios.

Once we have obtained the measures on the error according to N_r, we apply the minimum difference method presented in Section V-A to choose N_r^{opt}, i.e., the optimal value of N_r. Fig. 6 shows the N_r^{opt} value obtained for each scenario.

We notice that when the UE is mobile (scenarios MC and MA), our algorithm estimates that a significantly larger number/frequency of CQI reports is needed compared with the static scenarios (SLD, SMD, and SSD). For the latter, the error reduces as the distance decreases, and thus the optimal N_r decreases as well.

b) LSTM method results: Here we present the results of the application of the forecasting method based on LSTM, which we have implemented in MATLAB [26]. After several tests, we selected in this LSTM network the Adam [27] solver for training, using 200 epochs. We argue for this choice by the fact that this configuration gave more precision at the output. We trained the model on sequences of d_{PRB} values that correspond to 200 collected CQI samples during 200 ms periods. Then, following a monitoring period τ of $\tau = 200$ ms where an input sequence is collected, we perform forecasting of the d_{PRB} values for the next period $\tau + 1$. The considered forecasting period durations are: 5, 10, 20, 50, 100 and 200 ms, where for each duration we calculate the corresponding d_{PRB} errors, as shown in Fig. 7.

Similarly to the previous results, we notice that the error of the MC and MA and sometimes the SLD scenarios are the highest, followed by the SMD and SSD. In addition, we observe that the longer the prediction period, the higher the error. We attribute this to the fact that the prediction time goes far beyond the actual values recorded to make the prediction.

c) Comparison between the Optimal Difference method and LSTM: We recall that the methodology and metrics used to compare between the two methods are detailed in Section V-C. For the optimal difference method, a higher gain means fewer retrieved CQI reports during a prediction period of the same fixed duration as a monitoring period; the rest of the CQI values are generated as described in Section V-A. For LSTM, this means a longer prediction period where we generate a sequence of d_{PRB} values using the learned model and with the sequence of values of the last monitoring period as input; the duration of this prediction period is left to the system operator.

In Fig. 8, we draw the average normalized d_{PRB} error of the five considered scenarios against the normalized reporting gain. The latter is an expression of the number of

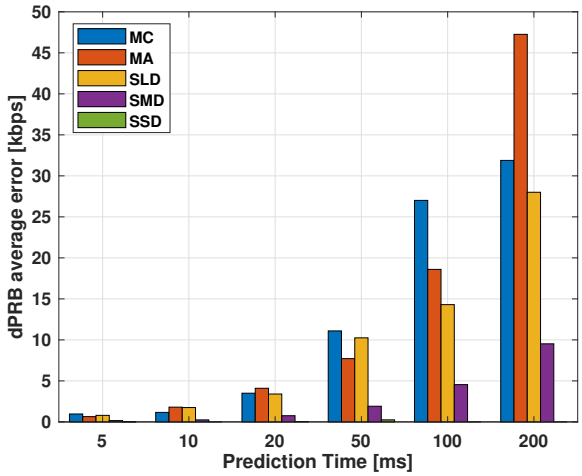


Fig. 7: d_{PRB} error vs. mobility state using LSTM.

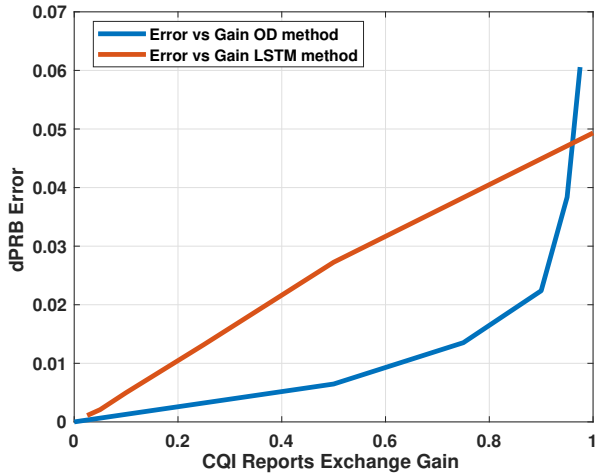


Fig. 8: d_{PRB} normalized error vs. gain.

CQI reports that are predicted (i.e., not actually retrieved) and thus represents the savings in terms of CQI monitoring traffic compared to the case where the default CQI collection takes place without any optimizations or prediction (as, e.g., in [9] and [14]). As per the definitions of Section V-C, for the LSTM method, when the number of predicted values equals the default number of CQI reports that would normally be collected during the period in question (in other words, when the prediction period has the maximum duration), the LSTM gain reaches 100%. On the other hand, for the optimal difference method, the gain depends on the optimal N_r value selected; the lower this value, the higher the gain.

We can observe that the optimal difference method performs better than LSTM in terms of error (up to 95% of gain). The error of the optimal difference method is smaller until reaching 95% of the gain; beyond this percentage, LSTM performs better. A d_{PRB} value predicted by the optimal difference method is always the same as the last actual value that corresponds to a real collected CQI sample. As the gain increases, the number of such samples decreases, which drives the estimation error up. In such high-gain conditions, the values predicted by the LSTM model can better capture the actual variation of real d_{PRB} ones. We should note that contrary to the LSTM method, which does not need to retrieve any actual CQI values during a forecasting period and can thus reach a gain of 100%, the optimal difference one, by design, always needs to retrieve *at least one* CQI value during its testing period (i.e., the number of CQI reports N_r cannot be zero) in order to consider it as the predicted CQI value for the rest of the interval when no CQI report will be collected.

B. RAN slicing with optimization

1) Data driven RAN slicing: First, we study the performance of the RAN slicing methods without the CQI reporting optimization (proposed in Section III). The proposed algorithm runs at the SO level. In this simulation, we considered two types of slices, i.e., URLLC and eMBB. Each slice is defined by the number of users, the required application data rate, the maximum latency (Lat_{max}) for URLLC, etc.

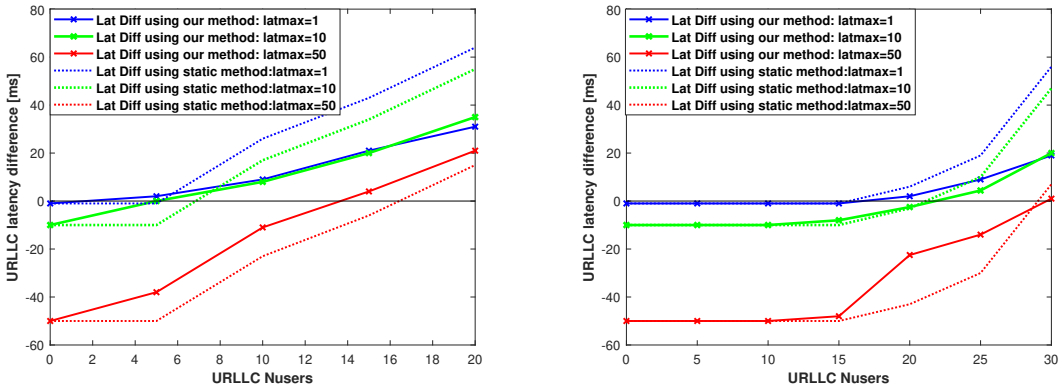
We simulated different scenarios, where we fixed the number of users of the eMBB slice ($N_{eMBBusers}$) to 5, while varying this number for the URLLC slice ($N_{uRLLCusers}$). We considered different channel qualities: (i) medium quality where the CQI varies from 7 to 9; (ii) good quality where the CQI varies from 13 to 15. Note that we simulated the case of only one BS and one SO. Table III presents the simulation parameter set in all scenarios.

TABLE III: Simulation parameters

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

In order to see the effectiveness of our solution, we compared it with the adopted method in [5], which uses a statically chosen percentage of PRBs corresponding to a fixed slice-dedicated bandwidth (SDB), and distributes it among the different slice users; in our tests, we considered a percentage of 50% of SDB per slice. This static radio resource partitioning corresponds to one of the approaches for network sharing considered by the 3GPP [28]. It is worth noting that the number of PRBs available is limited by the channel bandwidth. In our simulation, we selected to use a bandwidth of 5 Mhz, where 25 PRBs are available, in order to quickly saturate the channel and show the efficiency of our proposed solution. The only difference for higher bandwidths is the threshold beyond which our solution does not perform well.

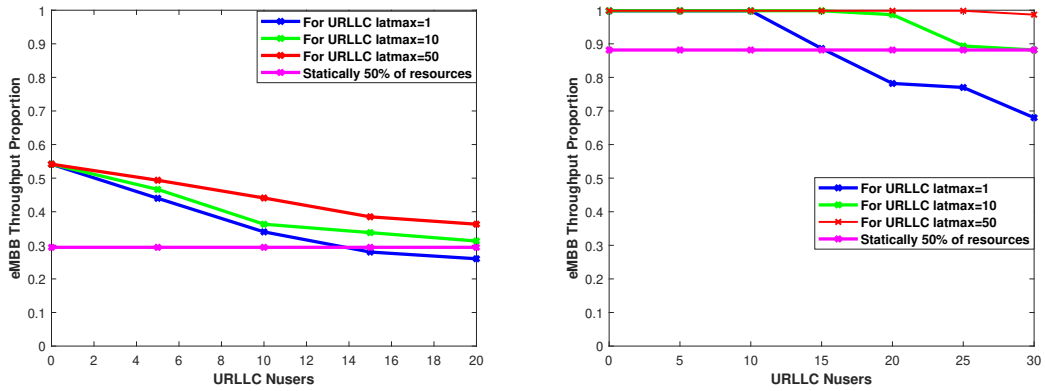
Note that the combined number of PRBs to be allocated to both URLLC and eMBB slices may exceed the capacity of the channel; hence, in this implementation, we applied a fair share of resources between slices, which has been calculated



(a) Medium channel quality.

(b) Good channel quality.

Fig. 9: (Lat-$Latmax$) vs. the number of URLLC users.



(a) Medium channel quality.

(b) Good channel quality.

Fig. 10: Throughput proportion vs. the number of URLLC users.

as follows. In order to see the effectiveness of our solution, we compared it with the adopted method in [5], which uses a statically chosen percentage of PRBs corresponding to a fixed slice-dedicated bandwidth (SDB), and distributes it among the different slice users; We first calculate the difference between the number of PRBs available and the number of PRBs requested for both slices as shown below:

$$\Delta = N_{PRBmax} - (N_{PRBuRLLC} + N_{PRBeMBB}).$$

Then, we reduce the same amount of PRBs ($\frac{|\Delta|}{2}$) from each slice in order to adapt it to the capacity of the channel.

Finally, we compute (a) for the URLLC slice: the difference Lat_{max}-Lat between (i) the latency obtained by allocating the number of resources obtained (Lat) using our method and the static method of 50% SDB, and (ii) the required latency Lat_{max}, as shown in Fig. 9a and 9b; and (b) for the eMBB slice: the proportion of the allocated throughput (using our method of resource sharing, and the static of 50% SDB) relative to the throughput required by this slice (Fig. 10a and 10b), in order to evaluate the accuracy of the proposed methods of estimating the radio resource d_{PRB} values required by each slice type dynamically.

Afterwards, we simulate the URLLC slice latency difference between Lat and Lat_{max}, for different numbers of $N_{uRLLCusers}$, and for two channel qualities, good and medium, as shown Fig. 9a and 9b respectively. We considered three service-level requirements (SLR) for the URLLC slice, which are three values of Lat_{max}: 1 ms, 10 ms and 50 ms.

Note that when Lat-$Lat_{max} < 0$, the latency required by the URLLC slice is respected. Otherwise, when this difference is positive, the obtained latency Lat no longer respects the slice requirements.

The simulation results here show that whatever the value of Lat_{max}, and for both channel qualities, our algorithm allows to keep the latency around Lat_{max} (the latency difference is negative), for up to a certain number of users. Therefore, we remark that there is a threshold on the number of URLLC users beyond which the latency exceeds Lat_{max}. These thresholds are: 2, 5 and 14 for the medium channel quality when $Lat_{max} = 1$ ms, 5 ms and 50 ms respectively, and 15, 22 and 29 for the good channel quality when Lat_{max} =1 ms, 5 ms

and 50 ms respectively. We explain the difference between these values by the fact that good channel quality allows to have higher N_{PRB} compared to medium channel quality, supporting more users in the URLLC slice.

Regarding the static PRB allocation method, where 50% of SDB is allocated to the slice (whatever the SLR of this slice), we remark that the differences between the latency obtained using the static method and the three Lat_{max} values considered, are always larger than the differences obtained using our method for each Lat_{max} respectively. Therefore, the static method is not optimal.

Furthermore, the throughput proportion obtained for the eMBB slice according to the number of the URLLC slice's users, for both considered channel qualities is shown in Fig. 10a and 10b. Here also, we note that there is a threshold on the number of users beyond which the performance of the slice degrades.

Our solution cannot guarantee 100% of the requested bandwidth for the medium channel quality. However, it guarantees the needed bandwidth until 10 and 25 users when Lat_{max} =1 ms and 50 ms, respectively, for the good channel quality. This is expected, as the URLLC users need more PRBs in the case of Lat_{max} =1 ms, which strongly affects the eMBB users. In addition, we observe that using a fixed number of PRBs, always ensures the same throughput (less than 100% of the total required), which is not optimal.

To summarize, the simulation results of the N_{PRB} estimation for the URLLC and eMBB slices clearly indicate the ability of our solution to slice the RAN resources and satisfy the heterogeneous requirements of both types of network slices when not exceeding a certain threshold on the number of users, due to the bandwidth limitations.

One of the main weaknesses of the encountered solution is the overhead that arises due to the frequent exchange of CQI values between the SO and eNBs.

2) Data-driven RAN slicing with optimization: To evaluate the efficiency and the impact of the optimized RAN slicing solution on the slice requirements, we have integrated the CQI reporting frequency reduction algorithms to our mechanisms.

In this simulation, we considered the same parameters, as shown in Table III. However, here each user has the real channel quality during the test period, using the real CQI measurements obtained from the CRAWDAD data set.

The simulated results in this part are based on the calculation of the throughput of the eMBB slice before and after the application of the CQI report reduction algorithms. Note that we have computed only the throughput of the eMBB slice, as the throughput assigned to each slice depends directly on the number of PRBs allocated to each slice. Thus we can show the optimization impact easily.

Fig. 11 illustrates the default throughput assigned to the eMBB slice before applying any mechanisms to reduce the frequency of CQI reporting and after applying the Optimal Difference method. The CQI report reduction algorithm was applied as follows: For each scenario, we calculated the optimal number of reports N_r^{opt} over a monitoring period, and then we applied it over the period that follows. Finally, we calculated the throughput corresponding to this N_r^{opt}.

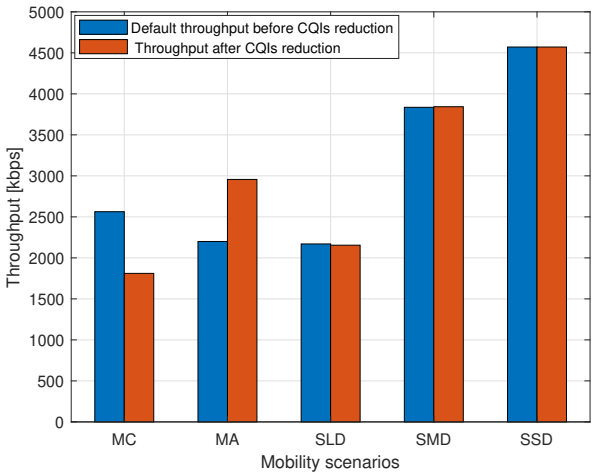


Fig. 11: Throughput before and after applying the Optimal Difference method to reduce the CQI reporting frequency, for different user mobility scenarios.

The results show that for the stable scenarios (SLD, SMD, and SSD), the throughput assigned to each slice was not impacted by reducing the CQI reporting frequency considering N_r^{opt}. A small change is noticed in the two mobile cases MC and MA. In the MC case, less throughput was assigned after reduction, while in the MA case, more throughput was assigned. This error of assigning less throughput to MC and more to MA depends on the CQI values (high, medium, or low) retrieved using N_r^{opt}.

The results of the CQI reporting rate reduction using LSTM are shown in Fig. 12. Here, as there is no optimal number or interval reduction that would show the impact of LSTM, we calculate the average percentage of the throughput error by report to the real throughput. The latter is calculated for each scenario and using all of the previously considered prediction intervals, which are: 200, 100, 50, 20 and 5 ms (i.e., the average error between these time intervals). The obtained results show that when user mobility increases, the LSTM cannot correctly predict CQI values. This has an impact on the slice requirements in terms of throughput since the error decreases when mobility decreases.

VII. Conclusion

In this paper, we devised a data-driven algorithm for sharing RAN resources among heterogeneous slices. The proposed algorithm computes and adjusts the radio resources needed by each running slice, using feedback on users' CQI. The proposed algorithm runs at the slice orchestrator level with very low complexity. Our results indicate the ability of the proposed algorithm to dynamically guarantee constraints of eMBB and URLLC slices when the number of active users stays below a certain threshold. In addition, to mitigate the overheads

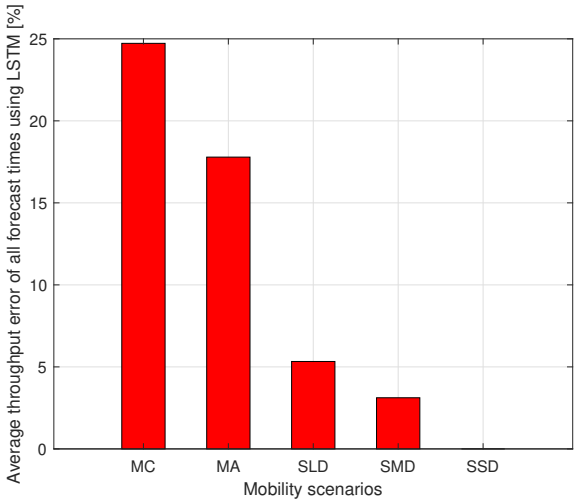


Fig. 12: Average throughput error across all considered forecasting period durations using LSTM, for different user mobility scenarios.

associated with frequent CQI monitoring, we proposed two predictive methods, i.e., *Optimal Difference* and *LSTM*-based forecasting. Our objective was to reduce the frequency of CQI report exchanges between base stations and the slice orchestrator while minimizing the error of our estimates of the achievable throughput when the CQI reporting frequency is reduced. Again, our simulation results demonstrate the positive impact of our CQI reporting optimizations by reducing the overhead while maintaining a precise prediction of RAN resources for the running network slices.

ACKNOWLEDGMENT

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

REFERENCES

[1] 3GPP, TS 23.501, "5G; System Architecture for the 5G System," v15.3.0, Release 15, Sept. 2018.

[2] K. B. Letaief, W. Chen, Y. Shi, J. Zhang, and Y. A. Zhang, "The Roadmap to 6G: AI Empowered Wireless Networks," in *IEEE Communications Magazine*, vol. 57(8), pp. 84–90, Aug. 2019.

[3] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Toward 6G Networks: Use Cases and Technologies," in *IEEE Communications Magazine*, vol. 58(3), pp. 55–61, Mar. 2020.

[4] W. Saad, M. Bennis, and M. Chen, "A Vision of 6G Wireless Systems: Applications, Trends, Technologies, and Open Research Problems," in *IEEE Network*, vol. 34(3), pp. 134–142, May/Jun. 2020.

[5] A. Ksentini, P. A. Frangoudis, A. PC, and N. Nikaein, "Providing low latency guarantees for slicing-ready 5G systems via two-level MAC scheduling," in *IEEE Network*, vol. 32(6), pp. 116–123, Nov./Dec. 2018.

[6] S. Bakri, P. A. Frangoudis, and A. Ksentini, "Dynamic slicing of RAN resources for heterogeneous coexisting 5G services," in *Proc. IEEE GLOBECOM*, 2019.

[7] A. Ksentini and N. Nikaein, "Toward enforcing Network Slicing on RAN: Flexibility and Resources abstraction," in *IEEE Communications Magazine*, vol. 55(6), pp. 102–108, Jun. 2017.

[8] S. Bakri, M. Bouaziz, P. A. Frangoudis, and A. Ksentini, "Channel stability prediction to optimize signaling overhead in 5G networks using machine learning," in *Proc. IEEE International Conference on Communications (ICC)*, 2020.

[9] P. L. Vo, M. N. H. Nguyen, T. A. Le, and N. H. Tran, "Slicing the Edge: Resource Allocation for RAN Network Slicing," in *IEEE Wireless Communications Letters*, vol. 7(6), Dec. 2018.

[10] H. Zhang, N. Liu, X. Chu, K. Long, A. Aghvami, and V. C. M. Leung, "Network Slicing Based 5G and Future Mobile Networks: Mobility, Resource Management, and Challenges," in *IEEE Communications Magazine*, vo. 55(8), Aug. 2017.

[11] A. Anand, G. de Veciana, and S. Shakkottai, "Joint Scheduling of URLLC and eMBB Traffic in 5G Wireless Networks," in *IEEE/ACM Trans. Netw.*, vol. 28(2), pp. 477–490, 2020.

[12] P. Caballero, A. Banchs, G. de Veciana, X. Costa-Pérez, and A. Azcorra, "Network Slicing for Guaranteed Rate Services: Admission Control and Resource Allocation Games," in *IEEE Transactions on Wireless Communications*, vol. 17(10), Oct. 2018.

[13] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia, and A. Banchs, "Mobile Traffic Forecasting for Maximizing 5G Network Slicing Resource Utilization," in *Proc. IEEE INFOCOM*, 2017.

[14] J. Salvat, L. Zanzi, A. G.Saavedra, V. Sciancalepore, and X. Costa-Pérez, "Overbooking Network Slices through Yield-driven End-to-End Orchestration," in *Proc. ACM CoNEXT*, 2018.

[15] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, "FlexRAN: A Flexible and Programmable Platform for Software-Defined Radio Access Networks," in *Proc. ACM CoNEXT*, 2016.

[16] L. Kleinrock, *Queueing Systems, Volume I: Theory*. New York, NY, USA: Wiley, 1975.

[17] U. C. Kozat and A. C. K. Soong, "On the Impact of Slicing Granularity on the Availability and Scalability of 5G Networks," in *Proc. IEEE ICC*, 2019.

[18] P. Castagno, V. Mancuso, M. Sereno, and M. Ajmone Marsan, "Closed form Expressions for the Performance Metrics of Data Services in Cellular Networks," in *Proc. IEEE INFOCOM*, 2018.

[19] I. Afolabi, T. Taleb, P. A. Frangoudis, M. Bagaa, and A. Ksentini, "Network Slicing-Based Customization of 5G Mobile Services," in *IEEE Network*, vol. 33(5), pp. 134–141, 2019.

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

[21] Latif, S., Rana, R., Younis, S., Qadir, J., Epps, J. (2018) Transfer Learning for Improving Speech Emotion Classification Accuracy. Proc. Interspeech 2018, 257-261, DOI: 10.21437/Interspeech.

[22] U. Cote-Allard, C. Latyr Fall, A. Drouin, A. Campeau-Lecours, C. Gosselin, K. Glette, F. Laviolette and B. Gosselin, Deep Learning for Electromyographic Hand Gesture Signal Classification by Leveraging Transfer Learning, ArXiv eprints, Jan. 2018.

[23] A. Phinyomark, S. Hirunviriya, C. Limsakul and P. Phukpattaranont, "Evaluation of EMG feature extraction for hand movement recognition based on Euclidean distance and standard deviation," ECTI-CON2010: The 2010 ECTI International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, Chiang Mai, Thailand, 2010.

[24] H. Cheng, Z. Xie, L. Wu, Z. Yu, and R. Li, "Data prediction model in wireless sensor networks based on bidirectional LSTM," in *EURASIP J. Wireless Com. Network*, 203, 2019.

[25] B. Koksal, R. Schmidt, X. Vasilakos, N. Nikaien, CRAWDAD dataset eurecom/elasticmon5G2019 (v. 2019-08-29).

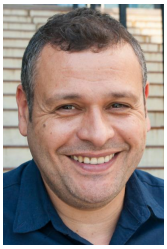
[26] Long Short-Term Memory (LSTM) in *Time Series Forecasting Using Deep Learning* MATLAB, Deep Learning Toolbox.

[27] D.P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. ICLR*, 2015.

[28] *Universal Mobile Telecommunications System (UMTS); LTE; Service aspects; Service principles*, 3GPP TS 22.101, v14.8.0, Release 14, Oct. 2017.

Sihem Bakri was a PhD student at the Department of Communication Systems, EURECOM, France (2017-2021), and received her PhD degree in Telecommunications from the Sorbonne University in 2021. Her doctoral thesis was an excellence grant from Institut Mines-Télécom, France. She received a Master's degree in Intelligent Telecommunication Systems from the University of Science and Technology Houari Boumediene (Algiers), in 2016. In 2017, she received another Master's degree in Telecommunications Techniques and Technologies from the University of Paris-Est Marne la Vallée. She is currently a postdoctoral researcher at Telecom Sud Paris of the Institut Mines Telecom. Her research interests include 5G networks, network slicing, network resource optimization, AI, reinforcement learning and mathematical modeling.

Pantelis A. Frangoudis is a post-doctoral researcher with the Distributed Systems Group, TU Wien, Austria. He has been a researcher with the Communication Systems Department, EURECOM, France (2017-2019), and with team DIONYSOS at IRISA/INRIA Rennes, France (2012-2017), which he originally joined under an ERCIM "Alain Bensoussan" post-doctoral fellowship. He has a Ph.D. (2012) in Computer Science from AUEB, Greece. His interests include mobile and wireless networking, network softwarization, edge computing, and Internet multimedia.



Adlen Ksentini Adlen Ksentini is a COMSOC distinguished lecturer. He obtained his Ph.D. degree in computer science from the University of Cergy-Pontoise in 2005, with a dissertation on QoS provisioning in IEEE 802.11-based networks. From 2006 to 2016, he worked at the University of Rennes 1 as an assistant professor. During this period, he was a member of the Dionysos Team with INRIA, Rennes. Since March 2016, he has been working as a professor in the Communication Systems Department of EURECOM. He has been involved in several national and European projects on QoS and QoE support in future wireless, network virtualization, cloud networking, mobile networks, and more recently on Network Slicing and 5G in the context of H2020 projects 5G!Pagoda, 5GTransformer, 5G!Drones and MonB5G. He has co-authored over 120 technical journal and international conference papers. He received the best paper award from IEEE IWCMC 2016, IEEE ICC 2012, and ACM MSWiM 2005. He has been awarded the 2017 IEEE Comsoc Fred W. Ellersick (best IEEE communications Magazine's paper). Adlen Ksentini has given several tutorials in IEEE international conferences, IEEE Globecom 2015, IEEEE CCNC 2017, IEEE ICC 2017, IEEE/IFIP IM 2017. Adlen Ksentini has been acting as TPC Symposium Chair for IEEE ICC 2016/2017, IEEE GLOBECOM 2017, IEEE Cloudnet 2017 and IEEE 5G Forum 2018. He has been acting as Guest Editor for IEEE Journal of Selected Area on Communication (JSAC) Series on Network Softwerization, IEEE Wireless Communications, IEEE Communications Magazine, and two issues of ComSoc MMTC Letters. He has been on the Technical Program Committees of major IEEE ComSoc, ICC/GLOBECOM, ICME, WCNC, and PIMRC conferences. He acted as the Director of IEEE ComSoc EMEA region and member of the IEEE Comsoc Board of Governor (2019-2020). He is the chair of the IEEE ComSoc Technical Committee on Software (TCS).

Maha Bouaziz has a PhD in Computer Science. Currently, she is working as a Teacher-Researcher at University of Strasbourg from september 2020. Before, she was in a postdoctoral position at EURECOM. Her current research focuses on wireless networks and innovation such as IoT, 5G, Network slicing, mobility management, Machine Learning,...