Slicing with Guaranteed Quality of Service in WiFi Networks

Matías Richart*†, Javier Baliosian*, Joan Serrat†, Juan-Luis Gorricho† and Ramón Agüero‡
*University of the Republic, Uruguay, †Universitat Politècnica de Catalunya, Spain, ‡University of Cantabria, Spain,
{mrichart, javierba}@fing.edu.uy, serrat@tsc.upc.edu, juanluis@entel.upc.edu, ramon@tlmat.unican.es

***Abstract*—Network slicing has recently been proposed as one of the main enablers for 5G networks. The slicing concept consists of the partition of a physical network into several self-contained logical networks (slices) that can be tailored to offer different functional or performance requirements. In the context of 5G networks, we argue that existing ubiquitous WiFi technology can be exploited to cope with new requirements. Therefore, in this paper, we propose a novel mechanism to implement network slicing in WiFi Access Points. We formulate the resource allocation problem to the different slices as a stochastic optimization problem, where each slice can have bit rate, delay, and capacity requirements. We devise a solution to the problem above using the Lyapunov drift optimization theory, and we develop a novel queuing and scheduling algorithm. We have used Matlab and Simulink to build a prototype of the proposed solution, whose performance has been evaluated in a typical slicing scenario.**

***Index Terms*—5G, Quality of service, Stochastic optimization, Wireless LAN, Wireless network slicing, Resource management.**

I. Introduction

Network Slicing is a new network paradigm developed within the context of recent 5G networks, which proposes the partition of the physical network infrastructure into multiple self-contained logical (or virtual) networks called slices. Using this paradigm, the infrastructure owners can allocate resources to service providers (*tenants*), creating dynamic and on-demand resource slices. The tenants have complete control over those resources, and they use them to satisfy their client demands. Also, 5G networks are designed to be multi-technology, and they are being deployed to work with already existing infrastructures. In this context, the available IEEE 802.11 (WiFi) infrastructure, which is massively deployed, can be used to leverage 5G capabilities. In this paper, we focus on the WiFi technology, for which slicing has not been thoroughly studied [1], regardless of its doubtless relevance.

There have been recent advances in standardized specifications by the 3GPP organization for 5G systems. In [2] the most relevant network slicing requirements are identified, describing that slices may be tailored to provide different functional requirements but also may have different performance requirements. In this work, we focus specifically on performance requirements, and we elaborate on how it could be implemented in WiFi Radio Access Networks (RANs). We define this slicing strategy as *Quality-of-Service Slicing (QoSS)*: slices supporting different services and ensuring their Quality of Service (QoS), regardless of the required resources. For example, a slice can be created to assure a minimum guaranteed latency or to provide a minimum guaranteed bandwidth to a given service. In particular, the approach followed is to consider the QoS requirement of a slice as the QoS guaranteed to each traffic flow[1] within the slice. We deem this approach as more interesting for a tenant who asks a QoS guarantee for its clients. This is consistent with current design definitions in 5G, where the concept of QoS flows is introduced [3]. However, there exist other approaches in the literature where performance guarantees are requested for the entire slice and not for each flow [4].

In this regard, achieving performance guarantees in wireless networks is challenging mainly because of the variability of wireless links' capacity and because of limited resources. In wireless communications, the capacity of the link depends on the *Signal-to-Interference-plus-Noise Ratio* (SINR) and the available bandwidth of the link. The SINR depends mainly on the transmission power, the attenuation of the signal while it crosses the wireless medium, the noise, and the interference. As lower is the SINR at a receiver, lower is the maximum data rate at which the receiver can decode a signal. Hence, for a given wireless communication, the transmission data rate is variable over time due to, for example, the distance between transmitter and receiver, the location of the interferers, or the obstacles in between communicating nodes. Besides, the available radio spectrum is not only limited by technology, but it is also regulated. This implies a bound on the bandwidth of the wireless channel that can be used and limits the possibilities to be increased, in contrast to the usual deployment of wired networks.

Consequently, to guarantee any performance requirements, a dynamic resource provisioning strategy is needed. In this work, we propose a slicing solution that considers the transmission time (*airtime*) as the resource to share, and it modifies the packet queuing and scheduling of the WiFi Access Points (APs), to control the airtime that is allocated to the different slices. In particular, the proposed solution seeks to provide a minimum guaranteed transmission bit rate and guaranteed maximum queuing delay at the AP. The solution is designed by formulating the guaranteed bit rate and bounded queuing-delay slicing problem as a *stochastic optimization problem*, where the channel conditions and the arrival rates are unknown stochastic processes. Our solution applies the *drift-plus-penalty* methods described in [5], which is based on the *Lyapunov Optimization Theory*. This method permits to obtain an equivalent deterministic problem, which provides an approximate solution.

We perform this by extending our previous work on airtime slicing [6], [7] and guaranteed bit rate slicing [8] in WiFi APs. The main differences of this work with [8] are threefold:

1) We add a new QoS guarantee to the system model, a delay bound to every packet of a flow.

[1]A traffic flow is a stream of packets sent between a given source and destination. For example, a flow in an IP network is identified by the source and destination IP addresses and the source and destination ports.

2) We also add a capacity limit to the slices, to restrict the slice's resource consumption.
3) The solution is evaluated on an extended scenario, with slices with different QoS requirements, and flows with different traffic patterns.

The rest of the paper is organized as follows: in Section II, we review the state of the art in wireless slicing and scheduling mechanisms; in Section III we describe our system and QoS guarantees models. Section IV depicts the problem formulation as a stochastic optimization problem, and in Section V we propose a solution based on the Lyapunov Optimization Theory. In Section VI, we propose a mechanism to guarantee isolation when the channel conditions do not permit to guarantee the required performance, while in Section VII we discuss several considerations concerning the implementation of our solution for WiFi technology. In Section VIII we present an experimental evaluation of our solution. Finally, in Section IX we conclude the paper, identifying some aspects that will be tackled in our future research.

II. Background and Related Work

A. *Quality of Service in WiFi Networks*

Even though network slicing is a rather new concept, the problem of providing Quality of Service in IEEE 802.11 networks has been thoroughly studied during the last twenty years. In this sense, although in radio access networks the most predominant traffic is downlink (from the AP to the stations), most of the research on this subject has focused on providing QoS on the uplink (the transmissions from the stations to the AP), or between stations [9]. This is because WiFi networks have been traditionally seen as an extension of the wired local area network, and not as an Internet access alternative. Moreover, in many of those previous works, QoS provisioning is based on service differentiation and prioritization, but not on performance guarantees. Regarding bit rate (or throughput) guarantees for traffic from stations to the AP, the works of Banchs et al. [10], [11] are worth mentioning. They foster throughput guarantees utilizing access management schemes, controlling the Contention Window (CW) size. Differently, in our proposal, we only consider traffic from the AP to the stations, and our objective is to guarantee, at the AP, a minimum bit rate to each downlink flow of a slice.

Regarding delay control, one of the main components affecting packet delay in a network is queuing. Queuing delay is caused by the accumulation of packets in a queue, originated by the difference between the arrival and departure rates of packets and is affected by two main factors: the arrival and departure traffic rates and the formation of *persistent queues* because of traffic bursts. If the arrival rate is continuously higher than the departure rate, queues will build-up, and the delay would increase to infinity. Nevertheless, in more benevolent situations, queues can also introduce delays if, after a traffic burst, the queue is not completely emptied. This problem is called the *persistent full buffer problem* or *bufferbloat* [12], and is one of the main causes of queuing delay in the network edge. Moreover, in wireless networks, persistent queues can also be formed because of the burstiness of departures, since they use a shared medium and also due to the variable bit rate of the wireless transmissions. This makes queues to accumulate packets while waiting for a transmission opportunity and, consequently, to send them all in a burst when access to the medium is granted. Hence, controlling queuing delay implies keeping the length of the queues, yet taking into account all the above considerations. Notice that if the queue length is too short, the delay could be reduced, but at the cost of high packet loss and so throughput reduction. On the other hand, with too long queues, no packets would be lost, maximizing the throughput, but leading to excessive delays. In this regard, some approaches have dealt with bufferbloat in WiFi by controlling the queue length through Active Queue Management (AQM) techniques [13]. AQM mainly consists of intelligently dropping packets to avoid queues to grow indefinitely. In general, the main objective of AQM is to reduce network congestion by signaling (either implicitly, by dropping packets, or explicitly, by sending special packets) the sender to reduce the flow rate. Nevertheless, AQM has also a clear impact on queuing delay. However, AQM techniques have proved not to achieve good latency reductions in WiFi [14] because of the queuing at the lower layers of the WiFi stack. Therefore, the work in [15] implements a queue management mechanism at the queuing structure of the lower layers. The objective of this work is to reduce queuing delay while achieving airtime fairness. Nevertheless, it does not provide a bounded delay guarantee, nor it considers the slicing concept.

A different approach to provide QoS guarantees (particularly bit rate and delay) in wireless networks is to manage how the packets are scheduled for transmission. In the context of scheduling algorithms, in the last twenty-five years, the concept of *opportunistic scheduling* has been thoroughly studied to provide QoS guarantees [16]. These schedulers take advantage of physical layer information, such as the client channel capacity or local system information, such as the queue lengths. Nevertheless, all of the existing works have concentrated on theoretical proposals or cellular technologies, but, to the best of our knowledge, they have overlooked WiFi. In [17], a scheduler called *Modified Largest Weighted Delay First* (M-LWDF) is proposed to achieve average delay guarantees to traffic flows. The method is based on scheduling the packet with the largest product of queue length and transmission rate. A similar approach is proposed in [18], where users are prioritized based on an exponential formula using queue length and transmission rate. In [19], an approach is discussed where the scheduling function is based on a logarithmic expression of the queue lengths and the transmission rate. In [20], a scheduler is proposed where each QoS flow receives a fixed average throughput per slot, but other QoS objectives such as delay are not considered. However, most of those works consider priorities between different services, and the objective is to reduce the latency of the high priority flows. Conversely, in the slicing scenario, all slices are equally important but have different QoS requirements, which all have to be guaranteed simultaneously.

On the other hand, the work from Michael J. Neely on opportunistic scheduling [5], [21] shares some similarities with the works previously mentioned, but has some particular characteristics which make it more adequate for the slicing problem. The approach followed by Neely is to formulate the scheduling problem as a stochastic optimization problem, where the channel conditions and the arrival rates are unknown stochastic processes. Then, using the *Lyapunov Optimization Theory*, an optimization framework is proposed to solve this problem. The framework permits to obtain an equivalent deterministic problem, which provides an approximate bounded solution to the original stochastic one. This scheme provides the advantage that if the scheduling problem can be formulated as an optimization problem, a set of theoretic tools can be applied, and an approximate solution can be obtained. Even more, the distance between the optimal solution and the approximate solution is bounded by an adjustable parameter.

B. Slicing in WiFi Networks

Regarding slicing, some works have dealt with this particular issue in WiFi deployments. However, most of those works concentrate on what we call *Infrastructure-Sharing Slicing*. In this paradigm, the objective is to split and allocate network resources into slices, proportionally to a requested ratio of resources or priority, but performance guarantees are not considered. For example, the works [22], and [23] propose a framework based on Software Defined Network (SDN) to share a WLAN infrastructure through slicing. Moreover, during the last years, several works [24], [25], [26], [27], [6], [7], [28], [23] have proposed different solutions where Infrastructure-Sharing Slicing is achieved through the allocation of airtime ratios to each slice. Nevertheless, none of them consider QoS guarantees. A different approach is followed in [4], where the authors propose a scheduling mechanism with feedback control, to guarantee throughput ratios among slices. The proposal splits the total transmitted bytes of an AP into ratios, requested by the different slices. However, it also does not guarantee any performance to the slices. An extensive review of recent proposals for slicing in WiFi can be found in our previous paper [1].

C. Contributions

One of the main contributions of the work presented here is to consider the QoS Slicing problem as a stochastic optimization problem. We develop a system model and formulate the guaranteed bit rate and bounded queuing delay slicing problem as a stochastic optimization problem. After applying the Lyapunov Optimization Theory, we were able to build a very efficient scheduling algorithm from the obtained solution. However, the application of this approach to WiFi technology would not be possible without two other crucial mechanisms, developed in this work, which stand out as the main differences of our proposal with previous works on opportunistic scheduling.

First, given that scheduling in WiFi is not based on time slots, we needed to develop a mechanism to appropriately adapt the proposed scheduler. For this, we extended the approach originally introduced in [7]. It is based on a system that mimics a time-slotted solution, and it also provides feedback on the consumed airtime, which is crucial to calculate the actual channel capacity. Secondly, as the adopted theory does not consider cases when there is not a feasible solution (lack of resources), we introduce in this work the design and implementation of a mechanism to detect and correct unfeasible situations. In WiFi, because of the transmission medium characteristics, unfeasible cases may emerge during the scheduling process. Therefore, we have implemented a solution to tackle this issue (see Section VI). Furthermore, we use the queuing model proposed in our previous work, [7], which considers the particularities of the hardware behavior to avoid queue buildup at lower layers and to allow packet aggregation.

Finally, the main differences of the proposal discussed herewith from previous works on WiFi slicing is that it does not modify or tamper low-level MAC parameters, neither it needs feedback from the medium nor the stations to achieve the required allocation. It only requires information on the consumed airtime, which can be obtained from the hardware driver.

III. System Model

As already mentioned, our objective is to implement *QoS Slicing* on WiFi APs by dynamically allocating the necessary resources to the different slices. We devise the problem of guaranteeing a minimum bit rate to each client of a slice jointly with providing an upper bound on the queuing delay, as a dynamic resource allocation problem that can be optimized. For this, in our approach, we only consider downlink traffic, i.e., traffic from the AP to the clients. In this section, we define the network and system model to be used in the optimization problem formulation.

A. Dynamic Airtime Allocation

Regarding practical considerations, we assume that the traffic arrival rate to the different slices and clients is an unknown stochastic process. Moreover, we also assume that the AP is operating in a stochastic environment, which is difficult to model, so we consider it unpredictable and uncontrollable. In particular, given the medium access control of WiFi, we assume the existence of interference and congestion in the wireless medium, which may cause transmission waiting times as well as the possibility of packet collisions and/or packet losses. This may trigger the WiFi back-off procedure and packet retransmission, thus resulting in longer transmission times. In this scenario, to tackle all the wireless channel uncertainties mentioned above, we adopt the airtime allocation mechanism from our previous work, [7]. The approach followed is to implement dynamic allocation based on airtime control by modifying the *Adaptive Time-Excess Round Robin* (ATERR) algorithm presented in [7]. Briefly, ATERR follows a round-robin scheduling based on a given *quantum* of time. The quantum is a configurable parameter that controls how much airtime is allocated to each client in a round. When a packet is transmitted, the difference between the airtime consumed by the packet and the quantum is registered in a *time-excess* variable. Packets from the scheduled client are transmitted until the excess surpasses the quantum value and then, the algorithm moves to the next queue in a round-robin manner. Given that in WiFi the transmissions are made in frames, it is very likely that the size of a quantum does not exactly match a given number of frames. Then, in ATERR, the additional time consumed in one assignment is decremented from the next round.

This airtime resource allocation strategy followed by ATERR implements static allocation in that it just fulfills the allocation requests it receives. It allocates a fixed amount of resources only based on the requested ratio of each slice, which, given the characteristics of the wireless medium, makes it impossible to guarantee any performance metric. In other words, it does not consider the possibility of adapting the resource allocation to the different slices based on the achieved throughput or delay of the slice's traffic. Therefore, to be able to implement slices that guarantee some performance requirements, it is necessary to dynamically allocate airtime based on the current channel conditions. Moreover, with an airtime-based allocation, all the possible variations of the unknown environment are accounted in the airtime, providing an exact measurement of the time consumed in a transmission (see [7]). Hence, all the unknown variables from the medium that may affect our system are contemplated in the allocated airtime.

Accordingly, in this work, the approach is to modify ATERR to use the same fixed quantum size for every client. Then, it becomes possible to have a slotted transmission, with the difference that slots are of variable length, but leveraging a mechanism that yields, in the long run, the correct airtime assignment. In this sense, for a QoS Slicing solution, we need to develop a mechanism that selects the client to schedule for transmission on each time slot. To be able to perform this scheduling strategy, we also adopt the queuing structure proposed in [7] consisting of a queue per client and per slice. The scalability of this approach has been partially evaluated in [7], where a scenario with 30 queues is successfully tested. Furthermore, we consider that the proposed scheduling scheme would not hinder the system scalability, since the buffer capacity at the AP would likely be a much more limiting factor.

Finally, given the previous system model, let us consider the following parameters for a given AP:

- Let $\mathcal{S}$ be the set of slices instantiated in the AP, such that $\mid \mathcal{S} \mid$ is the number of slices defined in the AP.
- Let us consider that the slice $s \in \mathcal{S}$ serves a set of clients $\mathcal{N}_s$ where $\mid \mathcal{N}_s \mid$ is the total number of clients in slice s. We identify a client $n \in \mathcal{N}_s$ with the couple (n, s).
- K_s is the minimum bit rate requirement, which must be guaranteed to every client of the slice s.
- H_s is the slice s capacity limit, given as a ratio of the total resources.
- $A_{n,s}(t)$ is the arrival rate of the client n of slice s in time slot t. This rate is dynamic and evolves with time, but we assume it is invariant within a time slot.
- $\mathbb{C}_{n,s}(t)$ is the channel capacity between the AP and the client n of slice s in time slot t. This capacity is variable, and it depends on the channel conditions, but we assume to be invariant within a time slot. We also assume this is not the maximal theoretical capacity but a measured capacity which includes all the possible delays introduced by the unknown wireless environment.

From this time forth, because of space constraints, when a formulation applies for all clients of all the slices, we omit the expression $\forall s \in \mathcal{S}$, $\forall n \in \mathcal{N}_s$ when possible.

B. Bit Rate Modelling

As previously stated, one of the proposed objectives of our QoS Slicing approach is to guarantee a minimum bit rate to each client of a slice. In this regard, the problem is to find a control algorithm that decides how the AP should schedule the transmissions to the different clients to guarantee that each of them receives the minimum bit rate ensured by the corresponding slice.

Therefore, we first clearly define how we consider these bit rate guarantees. Let us first observe that the bit rate obtained by a client mostly depends on two factors: the channel capacity and the amount of time the AP transmits to that client. Therefore, we have that the bit rate to a client n of slice s, in time slot t, is given by:

$$R_{n,s}(t) = \mathbb{C}_{n,s}(t) \times x_{n,s}(t) \tag{1}$$

where $x_{n,s}(t)$ is the proportion (or ratio) of the time slot t assigned for transmitting to client n of slice s. Note that $\mathbb{C}_{n,s}(t)$ and $R_{n,s}(t)$ are respectively the channel capacity and the bit rate measured in bits per slot (for a generic time slot size). Therefore, $R_{n,s}(t)$ also represents the amount of data transmitted to client n of slice s in time slot t. Also, note that from the previous model description we have defined that the scheduler will assign the entire slot to just one client, then we have that $x_{n,s}(t) \in \{0, 1\}$ The problem to solve is thus finding, for all time slots t and all clients, the assignments $x_{n,s}(t)$ that satisfy the slice requests.

In this proposal, we consider slice requests consisting of a minimum average bit rate to be assured to each client of the slice. Therefore, the problem is to guarantee that the average bit rate of each client $(R_{n,s})$ would be within an interval of the requested average bit rate K_s. In other words, we propose an approach based on three parameters that define the slice's request: K_s, Δ_s and W_s. A slice tenant requests a minimum bit rate identified by K_s (in this work, we assume the bit rate measured in bits per slot, but the agreement can be defined in any metric previously settled), being what the tenant expects to be guaranteed by the provider. Also, to provide flexibility to the provider, it is possible to define a *tolerance* Δ_s, which measures the possible maximum deviation from the expected minimum bit rate. This parameter is also measured in bits per slot. Last, W_s is a time window over which the average bit rate is computed and where the minimum bit rate plus the deviation must be guaranteed. Hence, it is required that in a time window of size W_s every client n of the slice s receives, on average, a bit rate in the interval $(K_s - \Delta_s,\ K_s + \Delta_s)$.

Nevertheless, as already mentioned, $R_{n,s}(t)$ is a random (or stochastic) process, because the channel capacity varies randomly, depending on several factors. Accordingly, for our optimization problem formulation, where the objective is to find a resource allocation policy which can guarantee a minimum average bit rate, we consider the expected time average of the bit rate[2], defined as:

$$\overline{R}_{n,s} = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{R_{n,s}(\tau)\} \tag{2}$$

$$= \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\mathbb{C}_{n,s}(\tau) x_{n,s}(\tau)\} \tag{3}$$

Note that considering the expected average rewards is a classical utility function in infinite-horizon stochastic decision problems [30]. The objective is that the obtained solution converges to the required bit rate to respect the requested average bit rate within the indicated time window W_s.

C. Delay Modelling

As a delay guarantee, we propose to consider an upper bound on the delay of every packet. To achieve this objective, we adapted the approach described in [21], where the solution jointly manages the queue and the scheduler[3]. In summary, our proposed model manages two decision parameters (on each time slot) to guarantee that the delay of each packet is below a given threshold:

- select the next queue to schedule for transmission,
- drop packets from the head of the queues (the number of packets to drop is part of the decision).

[2]We refer the reader to the supplementary material [29] where it is shown that the proposed solution with this bit rate modeling approach yields the requested slice's guarantees under some necessary conditions.

[3]The work in [21] considers the possibility of rejecting incoming packets from the upper layers. In our solution, we disregard this possibility, as this would move the delay problem to the upper layers.

As already analyzed in Section II, one major factor in the queuing delay is the queue length. Hence, it becomes necessary to model the queue dynamics to consider its length on the problem formulation. Considering the possibility of dropping packets at the head of the queues, the queues' dynamics can be expressed as:

$$Q_{n,s}(t+1) = [Q_{n,s}(t) - R_{n,s}(t) - D_{n,s}(t)]^+ + A_{n,s}(t) \quad (4)$$

where $[\cdot]^+ = \max\{\cdot, 0\}$ and where $R_{n,s}(t)$, $D_{n,s}(t)$ and $A_{n,s}(t)$ is the amount of data transmitted, dropped and received, respectively, in slot t.

1) Persistent-Service Queues: To achieve bounded delay guarantees, we consider *ϵ-persistent service queues* into the problem. These queues are virtual, and they do not represent real network queues, but add new constraints to the problem to assure delay guarantees. These new virtual queues are defined by the following update equation for each client n of every slice s:

$$Z_{n,s}(t+1) = \begin{cases} [Z_{n,s}(t) + \epsilon_{n,s} - R_{n,s}(t) - D_{n,s}(t)]^+ & \text{if } Q_{n,s}(t) > 0 \\ [Z_{n,s}(t) - D_{n,s}(t) - R^{max}_{n,s}]^+ & \text{if } Q_{n,s}(t) = 0 \end{cases} \quad (5)$$

where $\epsilon_{n,s}$ are pre-defined constants.

In [29] we demonstrate that bounded delay is guaranteed by any control algorithm that maintains the size of both queues $Q_{n,s}(t)$ and $Z_{n,s}(t)$ bounded by finite maximums: $Q^{max}_{n,s}$ and $Z^{max}_{n,s}$, respectively. Even more, a bound for the delay is given by the constant $W^{max}_{n,s} = \left\lceil \frac{Q^{max}_{n,s} + Z^{max}_{n,s}}{\epsilon_{n,s}} \right\rceil$. Intuitively, the ϵ-persistent service queue allows having a virtual queue that always has "incoming traffic", so guaranteeing a bound on its length, jointly with the real data queues, permits to have an upper bound on the delay.

Then, with this approach, the guaranteed bounded delay problem is transformed into a problem of bounding queues. Therefore, the QoS Slicing formulation we are developing needs to include this requirement of queue bounds to bound the delay.

D. Slice Capacity Limit

Given that a fundamental aspect of network slicing is to provide isolation between slices when resources are scarce, our model includes the possibility to define a limit H_s on the relative allocation of resources to a slice s. In the context of our work, this parameter represents an upper bound on the average airtime consumption of a slice. For example, a slice may be limited to use only, on average, a third of the total available airtime at a particular AP. However, to use resources efficiently, we regard this parameter as a soft limit, in the sense that it could be violated when more airtime could be used without affecting other slices.

This limit can be negotiated with the slice tenant as part of the QoS agreement, depending on the required service or the type of users and/or traffic of the slice. This would provide the tenant an extra guarantee that in the event of having scarce resources at the AP, if the capacity limit is not surpassed, its QoS guarantees will be respected.

Let us consider $X_s(t) = \sum_{n \in \mathcal{N}_s} x_{n,s}$ as the airtime ratio allocated to slice s on time slot t. Then, following the expected average approach used in this model, we define the expected time average allocated airtime ratio to a slice s which will be limited by H_s as:

$$\overline{X}_s = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{X_s(\tau)\} \quad (6)$$

IV. Problem Formulation

Based on the previously described model, the problem is to find, for every slot t, the assignment vector, $\boldsymbol{x}(t) = [x_{n,s}(t) \mid s \in \mathcal{S},\ n \in \mathcal{N}_s]$, which guarantees that all slices' requests are satisfied. Therefore, we can formulate an optimization problem to find the resource allocation and drop decisions for maximizing the average expected total throughput of the AP and respecting the bit rate and airtime limits constraints.

Considering a system where the queues are stable, the throughput (measured in bits per slot) in a time slot t can be expressed as the amount of arrived data minus the amount of data dropped in slot t. We thus define our throughput function as:

$$u_{n,s}(t) = A_{n,s}(t) - D_{n,s}(t), \quad (7)$$

and its expected time average as:

$$\overline{u}_{n,s} = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{u_{n,s}(\tau)\}. \quad (8)$$

However, to consider fairness in the dropping decisions, we define the following fair utility function, which approximates proportional fairness (as suggested in [31]):

$$\phi(\boldsymbol{u}(t)) = \sum_{s \in \mathcal{S}} \sum_{n \in \mathcal{N}_s} \log(1 + \omega u_{n,s}(t)). \quad (9)$$

for some constant $\omega > 0$ and where $\boldsymbol{u}(t)$ is the vector of throughputs. Note that with this utility function, we are indeed minimizing the packet drops in relation to the arrival rates.

Then, we can formulate a *stochastic optimization problem* that maximizes the average expected total throughput subject to the bit rate, queue stability and airtime consumption constraints:

$$\begin{aligned} \underset{\boldsymbol{x},\boldsymbol{D}}{\text{maximize}} \quad & \phi(\overline{\boldsymbol{u}}) & (10) \\ \text{subject to} \quad & \overline{R}_{n,s} \geq K_s, & (11) \\ & \overline{X}_s \leq H_s, & (12) \\ & \overline{Q}_{n,s} < \infty, & (13) \\ & \overline{Z}_{n,s} < \infty, & (14) \\ & 0 \leq u_{n,s}(t) \leq A^{max}_{n,s}, & (15) \\ & x_{n,s}(t) \in \{0,1\}, & (16) \\ & \sum_{s \in \mathcal{S}} \sum_{n \in \mathcal{N}_s} x_{n,s}(t) \leq 1, & (17) \\ & 0 \leq D_{n,s}(t) \leq D^{max}_{n,s}. & (18) \end{aligned}$$

where

$$\phi(\overline{\boldsymbol{u}}) = \sum_{s \in \mathcal{S}} \sum_{n \in \mathcal{N}_s} \log(1 + \omega \overline{u}_{n,s}). \quad (19)$$

In this optimization problem the objective is to find the transmission airtime assignments $x_{n,s}(t)$ and the dropping decisions $D_{n,s}(t)$ to maximize the total average expected throughput. Note that $\boldsymbol{x} = \{\boldsymbol{x}(1), ..., \boldsymbol{x}(t), ...\}$ and $\boldsymbol{D} = \{\boldsymbol{D}(1), ..., \boldsymbol{D}(t), ...\}$ are the vectors of assignment and dropping decisions for all clients in all slots.

Constraint (11) considers the minimum average expected bit rate K_s of each slice. It represents, in fact, a set of constraints, since there is a bit rate requirement for each client of each slice. Constraint (12) limits the average airtime ratio assigned to each slice to its capacity limit H_s. Constraints (13) and (14) are the stability conditions for the packet and the ϵ-persistent service queues, respectively. Constraint (15) guarantees that the objective function is non-negative, by not dropping more packets than those that had arrived. Both (16) and (17) ensure that assigned resources do not surpass the available ones, by limiting the possible values of the $\boldsymbol{x}$ variables. Finally, constraint (18) limits the amount of data to drop at each time slot.

Note that in the problem formulation we have assumed a known maximum on the arrival rate per slot, $A_{n,s}(t) \leq A_{n,s}^{max}$ and we have also consider bounds on the dropping decisions, $0 \leq D_{n,s}(t) \leq D_{n,s}^{max}$, so that $D_{n,s}^{max}$ is the maximum amount of data that can be dropped in one slot. Finally, it is also important to note that there exists a maximum transmission rate given by $R_{n,s}^{max} = \mathbb{C}_{n,s}^{max}$ (the maximum capacity of the channel).

V. Proposed Solution

Our proposal consists of solving the previous stochastic problem by applying the *drift-plus-penalty* method described in [5]. This method allows us to build a new deterministic problem, which provides an approximate solution to the original one. Even more, such a solution can be made arbitrarily close to the optimal one, but with a trade-off on how constraints are fulfilled. For the proposed solution, we assume that the problem is feasible. That is, there exists an airtime allocation that satisfies all the problem constraints. Namely, we assume that slices' bit rate and delay requests can be fulfilled with the available resources. We argue that this is a valid assumption, as it is possible to have a previous mechanism of slice access control and a procedure to adapt slices when more resources than those available are needed. In particular, in Section VI, we develop a mechanism to deal with unfeasible situations.

A. Problem Transformation

Since problem (10)-(18) consists of the optimization of a concave non-linear function of time averages, we need to transform it, before applying the drift-plus-penalty method. This problem differs from the problem formulation needed to apply the method as it involves the optimization of a *function* of time averages. As the proposed function $\phi(\boldsymbol{u}(t))$ is not linear, in general, it is not the same to maximize a time average of a function than to maximize a function of time averages. Therefore, we follow the auxiliary variable technique described in [5], to adapt the formulated problem into a traditional time average optimization problem.

With this technique, the stochastic network optimization problem (10)-(18) can be transformed using a vector of auxiliary variables $\boldsymbol{\gamma}(t) = [\gamma_{n,s}(t) \mid s \in \mathcal{S},\ n \in \mathcal{N}_s]$, which are chosen at every slot, according to the constraints $0 \leq \gamma_{n,s}(t) \leq A_{n,s}^{max}$. The modified problem is therefore:

$$\begin{aligned}
&\underset{\boldsymbol{x},\boldsymbol{D}}{\text{maximize}} && \overline{\phi(\boldsymbol{\gamma})} && (20)\\
&\text{subject to} && (11)-(14), && (21)\\
& && \overline{\gamma}_{n,s} \leq \overline{u}_{n,s}, && (22)\\
& && 0 \leq \gamma_{n,s}(t) \leq A_{n,s}^{max}, && (23)\\
& && (16)-(18). && (24)
\end{aligned}$$

where

$$\overline{\phi(\boldsymbol{\gamma})} = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\left\{ \sum_{s\in\mathcal{S}} \sum_{n\in\mathcal{N}_s} \log(\gamma_{n,s}(\tau)) \right\} \tag{25}$$

Intuitively, we can explain the previous transformation as follows. First, it is worth noting that the original constraints are a subset of the new ones, so any solution to the transformed problem will also satisfy the original constraints. Suppose we have decisions $\boldsymbol{x}^*(t)$ and $\boldsymbol{D}^*(t)$, which are a solution to the original problem. Let $\overline{\boldsymbol{u}}^*$ be the expected utilities obtained by the clients under those decisions, which yield a maximum utility value $\phi(\overline{\boldsymbol{u}}^*) = \phi^{opt}$. Then, we can build a solution to the transformed problem with the same $\boldsymbol{x}^*(t)$ decisions, selecting $\boldsymbol{\gamma}(t) = \overline{\boldsymbol{u}}^*$ for all t. Note that this solution satisfies constraint (22), as we enforce equality, and new constraint (23) equals (15). As $\boldsymbol{\gamma}(t) = \overline{\boldsymbol{u}}^*$ is enforced for all t, we have that $\overline{\phi(\boldsymbol{\gamma})} = \phi(\overline{\boldsymbol{u}}^*) = \phi^{opt}$. Hence, we have a solution to the transformed problem with optimal value ϕ^{opt}, which is also a solution to the original problem. Therefore, a solution to the transformed problem ensures that the constraints of the original problem are satisfied, and it obtains a utility that approximates the original problem utility, as constraint (22) is forced to equality. In our supplementary material [29], we show the distance of the obtained solution to the optimum.

B. Application of the Drift-Plus-Penalty Approach

To solve the problem (20)-(24) using the drift-plus-penalty method, we first transform the constraints into queue stability problems. For constraints (11), (12) and (22) we define *virtual queues*, with update equations:

$$G_{n,s}(t+1) = [G_{n,s}(t) - R_{n,s}(t) + K_s]^+ \tag{26}$$

$$U_s(t+1) = [U_s(t) + X_s(t) - H_s]^+ \tag{27}$$

$$Y_{n,s}(t+1) = [Y_{n,s}(t) + \gamma_{n,s}(t) - u_{n,s}(t)]^+ \tag{28}$$

As already mentioned for the ϵ-persistent queues, these queues are virtual and so do not represent real network queues. Intuitively, they can be seen as queues that accumulate the difference between the required and the obtained performance. By ensuring stability on these three queues and on the already described $Q_{n,s}(t)$ and $Z_{n,s}(t)$ queues, we guarantee that the constraints (11)-(14) and (22) are met[4].

Then, the drift-plus-penalty strategy consists of minimizing all these queue's backlogs, as well as minimizing a utility function called *penalty*. As in our case, we have a reward maximization rather than a penalty minimization; we consider the opposite of our utility function $\phi(\boldsymbol{\gamma}(t))$ as a penalty. Let us consider $\boldsymbol{\Theta}(t) = [\boldsymbol{Q}(t), \boldsymbol{Z}(t), \boldsymbol{G}(t), \boldsymbol{U}(t), \boldsymbol{Y}(t)]$ as the concatenated vector of queue backlogs. Then, a *Lyapunov function* can be defined as a measure of the length of all the queues:

$$L(\boldsymbol{\Theta}(t)) = \frac{1}{2}\sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}_s} G_{n,s}(t) + \frac{1}{2}\sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}_s} Q_{n,s}(t) + \frac{1}{2}\sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}_s} Z_{n,s}(t) + \frac{1}{2}\sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}_s} Y_{n,s}(t) + \frac{1}{2}\sum_{s\in\mathcal{S}} U_s(t) \tag{29}$$

[4]We refer the reader to our supplementary material [29] for a complete demonstration of this fact.

From this definition it is introduced the *one-slot conditional Lyapunov drift* $\Delta(\boldsymbol{\Theta}(t))$ as:

$$\Delta(\boldsymbol{\Theta}(t)) = \mathbb{E}\{L(\boldsymbol{\Theta}(t+1)) - L(\boldsymbol{\Theta}(t))|\boldsymbol{\Theta}(t)\} \tag{30}$$

This difference (drift) represents the expected change in the Lyapunov function in one slot, given that the state in slot t is $\boldsymbol{\Theta}(t)$.

Accordingly, the strategy consists on minimizing on each time slot t the following expression:

$$\Delta(\boldsymbol{\Theta}(t)) - V\mathbb{E}\{\phi(\boldsymbol{\gamma}(t)) \mid \boldsymbol{\Theta}(t)\} \tag{31}$$

where V is a non-negative constant that will affect the trade-off between the drift and the penalty. Then, the drift-plus-penalty method comprises the minimization of the following upper bound of the previous expression:

$$\begin{aligned} &- V\mathbb{E}\{\phi(\boldsymbol{\gamma}(t)) \mid \boldsymbol{\Theta}(t)\} + \sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}_s} Y_{n,s}(t)\mathbb{E}\left\{\gamma_{n,s}(t) \mid \boldsymbol{\Theta}(t)\right\} \\ &-\sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}_s}[G_{n,s}(t)+Q_{n,s}(t)+Z_{n,s}(t)]\mathbb{E}\left\{R_{n,s}(t) \mid \boldsymbol{\Theta}(t)\right\} \\ &+\sum_{s\in\mathcal{S}} U_s(t)\mathbb{E}\left\{X_s(t) \mid \boldsymbol{\Theta}(t)\right\} \\ &+\sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}_s}[Y_{n,s}(t)-Q_{n,s}(t)-Z_{n,s}(t)\mathbb{E}\left\{D_{n,s}(t) \mid \boldsymbol{\Theta}(t)\right\} \end{aligned} \tag{32}$$

To solve this minimization, we separate the problem into three different goals: (1) a minimization of the $\gamma_{n,s}$ terms; (2) a minimization of the $x_{n,s}$ terms; and (3) a minimization of $D_{n,s}$ terms.

First, we find the optimal auxiliary variables, by considering a fixed $Y_{n,s}(t)$:

$$\begin{aligned} \underset{\boldsymbol{\gamma}(t)}{\text{minimize}} \quad & -V\phi(\gamma_{n,s}(t)) + \sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}_s} Y_{n,s}(t)\gamma_{n,s}(t) \\ \text{subject to} \quad & 0 \le \gamma_{n,s}(t) \le A_{n,s}^{max}. \end{aligned} \tag{33}$$

We can find a closed-form solution for problem (33). First, we transform the single minimization problem into a multiple problem by minimizing each sum term (remember that $\phi(\gamma_{n,s}(t)) = \sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}_s}\log(1+\omega\gamma_{n,s}(t))$). So, for each slice $s \in \mathcal{S}$ and for each client $n \in \mathcal{N}_s$ we have:

$$\begin{aligned} \underset{\boldsymbol{\gamma}(t)}{\text{minimize}} \quad & -V\log(1+\omega\gamma_{n,s}(t)) + Y_{n,s}(t)\gamma_{n,s}(t) \\ \text{subject to} \quad & 0 \le \gamma_{n,s}(t) \le A_{n,s}^{max}. \end{aligned} \tag{34}$$

Finding the derivative and setting it equal to zero we obtain:

$$\gamma_{n,s}(t) = \frac{V}{Y_{n,s}(t)} - \frac{1}{\omega}. \tag{35}$$

Secondly, also for each slot t, we observe the values of the queues $G_{n,s}(t)$, $Q_{n,s}(t)$, $Z_{n,s}(t)$ and $U_s(t)$, and the current channel state $\mathbb{C}_{n,s}(t)$, to find the $\boldsymbol{x}(t)$ that solves:

$$\begin{aligned} \underset{\boldsymbol{x}(t)}{\text{maximize}} \quad & \sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}_s}[\mathbb{C}_{n,s}(t)(G_{n,s}(t)+Q_{n,s}(t)+Z_{n,s}(t)) \\ & - U_s(t)]x_{n,s}(t) \\ \text{subject to} \quad & \sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}_s} x_{n,s}(t) \le 1, \\ & x_{n,s}(t) \in \{0,1\}. \end{aligned} \tag{36}$$

Finally, for each slot t, we need to solve the following problem (also considering $Y_{n,s}(t)$, $Q_{n,s}(t)$ and $Z_{n,s}(t)$ as given constants):

$$\begin{aligned} \underset{\boldsymbol{x}(t)}{\text{minimize}} \quad & \sum_{s\in\mathcal{S}}\sum_{n\in\mathcal{N}_s}[Y_{n,s}(t) - (Q_{n,s}(t)+Z_{n,s}(t))]D_{n,s}(t) \\ \text{subject to} \quad & 0 \le D_{n,s}(t) \le D_{n,s}^{max}. \end{aligned} \tag{37}$$

It is straightforward to observe that the solution to this problem is given by:

$$D_{n,s}(t) = \begin{cases} D_{n,s}^{max} & \text{if } Q_{n,s}(t)+Z_{n,s}(t) > Y_{n,s}(t) \\ 0 & \text{otherwise} \end{cases} \tag{38}$$

As a consequence, we get a deterministic optimization problem that, at every slot t, calculates the control actions $x_{n,s}(t)$ and $D_{n,s}(t)$ by observing the backlogs of the real queues $Q_{n,s}(t)$, the virtual queues $Z_{n,s}(t)$, $G_{n,s}(t)$, $U_s(t)$ and $Y_{n,s}(t)$ and the random channel capacities of each client $\mathbb{C}_{n,s}(t)$. Note that the channel capacities and the queue backlogs on time slot t act as constants in the optimization problem. In our supplementary material [29], we provide a proof that this solution satisfies all constraints in (20)-(24), and that the obtained utility differs from the target utility by no more than B/V, which can be made arbitrarily small as V is increased. However, the bound over the time average queues' backlogs increases linearly with V. In the case of our QoS Slicing problem, this trade-off translates into a compromise between the optimal utility achieved and the satisfaction of the bit rate and delay guarantees.

C. Proposed Scheduling Algorithm

The previous solution provides a mechanism that, at every time slot, resolves an optimization problem and finds the airtime allocations that must be assigned to each client of every slice. It also calculates the necessary packet drops at each queue. As we have already discussed, this task of assigning transmission opportunities to the different clients is performed by the *Scheduler* of the AP. Hence, based on the previous analysis, we develop a scheduling algorithm that implements the proposed solution.

It is easy to observe that the problem (36) has the form of the *Knapsack Problem*, where each object or item (n, s) in time slot t gives a reward of $\mathbb{C}_{n,s}(t)(G_{n,s}(t) + Q_{n,s}(t) + Z_{n,s}(t)) - U_s$, where all items weight 1 and the maximum capacity is also 1. It is straightforward to note that the solution to this problem is the item with the highest reward.

Then, from the previous analysis, we design the scheduling algorithm shown in Algorithm 1. The scheduler only considers non-empty queues at each slot to ensure that resources are assigned beyond the capacity limit H_s only when free resources are available. The rationale is that, if a client has an empty queue, it implies that all offered traffic has been already served, and so, if there were other queues with traffic, they should be served although having a lower utility. Also, in the algorithm shown, we adapt the quantum-based airtime allocation from ATERR to be used for this specific case.

As can be seen in Algorithm 1, each iteration of the algorithm corresponds to a time slot where the traffic queue of the client (n, s) with the highest value of $\mathbb{C}_{n,s}(t)(G_{n,s}(t) + Q_{n,s}(t) + Z_{n,s}(t)) - U_s$ is assigned for transmission. Then, packets are dequeued and transmitted until the quantum is totally consumed. After the transmission has ended, each client's queue is checked to decide if any packet drops are necessary. Finally, all virtual queues are updated accordingly.

Algorithm 1: QoS Slicing Scheduler Pseudocode.

```
1  function Scheduler() is
       input : $V$, $Q_{n,s}$, $H_s$, $K_s$, $D_{n,s}^{max}$, $\gamma_{n,s}^{max}$, $\epsilon_{n,s}$, $\omega$
       output: Scheduling and drops on each slot $t$
2      /* Initialize queues */
3      foreach $s \in \mathcal{S}, n \in \mathcal{N}_f$ do
4          $Z_{n,s} \leftarrow 0; G_{n,s} \leftarrow 0; Y_{n,s} \leftarrow 0; U_s \leftarrow 0;$
5      end
6      while true do
7          /* Observe the current state */
8          foreach $s \in \mathcal{S}, n \in \mathcal{N}_f$ do
9              $\mathbb{C}_{n,s} \leftarrow getCapacity(n, s);$
10         end
11         /* Compute the vector of benefits $\boldsymbol{B}$ */
12         $\boldsymbol{B} = \mathbb{C} * (\boldsymbol{G} + \boldsymbol{Q} + \boldsymbol{Z}) - \boldsymbol{U};$
13         /* Find the client $i$ with the max benefit */
14         $i \leftarrow arg\,max\,\boldsymbol{B};$
15         /* Get the queue of client $i$ */
16         $queue \leftarrow GetQueue(queueList, i);$
17         /* Transmit packets for a quantum period */
18         while $queue.excess < 0$ do
19             $airtime \leftarrow transmitPacket(queue);$
20             $queue.excess \leftarrow queue.excess + airtime$
21         end
22         $queue.excess \leftarrow queue.excess - QUANTUM;$
23         foreach $s \in \mathcal{S}, n \in \mathcal{N}_f$ do
24             if $Q_{n,s} + Z_{n,s} > Y_{n,s}$ then
25                 $dropPackets(n, s, D_{n,s}^{max});$
26             end
27             /* Calculate the auxiliary variables */
28             $\gamma_{n,s} \leftarrow \min\{\frac{V}{Y_{n,s}} - \frac{1}{\omega}, \gamma_{n,s}^{max}\};$
29             /* Update queue backlogs */
30             $Z_{n,s} \leftarrow [Z_{n,s} - \epsilon_{n,s} - R_{n,s} - D_{n,s}]^+;$
31             $G_{n,s} \leftarrow [G_{n,s} - R_{n,s} + K_s]^+;$
32             $U_s \leftarrow [U_s + X_s - H_s]^+;$
33             $Y_{n,s} \leftarrow [Y_{n,s} + \gamma_{n,s} - u_{n,s}]^+;$
34         end
35     end
36 end
```

VI. A Mechanism for Guaranteeing Isolation

In the context of QoS Slicing in wireless networks, the isolation between slices and between clients within a slice is an important aspect to keep the agreed guarantees, regardless of the clients' behavior. In this sense, we envision two different cases that would produce isolation issues. One of them appears when the offered traffic of a client within a slice exceeds the agreed maximum bit rate, and it thus consumes resources from other clients or slices. The other case happens when more resources than available are needed to satisfy all the slices' performance requests.

Therefore, we propose to classify the possible isolation violations in two different types: *Excess of offered load*, and *Lack of resources*. A possible solution to prevent the first type is that the slicing architecture must control and limit the traffic to conform to the parameters in the corresponding slice agreement. Note that allowing this traffic to enter the queues violates the arrival rate limitation $A_{n,s}^{max}$ imposed by the solution. This may affect the algorithm behavior by abnormally increasing the queues' length, and it may also yield a packet drop increase. A simple yet practical solution to this issue is to perform *traffic shaping* on the incoming flows. Traffic shaping techniques are widely used as they are very useful for the correct operation of networks with constrained resources. In particular, several techniques for rate limiting purposes already exist [32]. Hence, in this section, we focus on the second type of isolation violation.

A. Proposed Solution for Guaranteeing Isolation

The isolation issue of *Lack of resources* emerges when there are not enough resources to provide the agreed guarantees to each client, thus affecting their performance. Although admission control mechanisms may prevent this from happening when instantiating new clients or slices into the devices, the channel conditions of a client might worsen after the initial connection, causing the scheduler to take resources from other slices to provide the agreed QoS. From the perspective of the optimization problem formulated in Section V, this issue generates a situation of unfeasibility. This means that there is no possible airtime allocation that can meet all the required constraints.

As described in Section III, in the proposed QoS Slicing model, the isolation between different slices is provided by setting a limit on the amount of resources that a slice can use. With this approach, two objectives are achieved: (1) the expected time average airtime usage ratio of each slice is limited so as to avoid consuming resources from other slices; and (2) if extra resources are available (because one or more slices use fewer resources than requested), these are distributed among slices that have traffic to be served. Nevertheless, as we need to provide guarantees to each flow of a slice, the behavior of a client or a traffic flow may affect other flows of the same slice. Therefore, the issue of isolation among flows within the same slice must also be tackled by the slicing mechanism. Our isolation proposal consists of integrating the isolation management to the scheduler described in Algorithm 1. We propose a solution in two stages:

- A monitoring stage, to detect isolation violations.
- An action stage, where actions are taken to move the system to a stable state.

1) Monitoring: In this stage, the evolution of the virtual queues is monitored to detect isolation issues. We recall that the virtual queues $G_{n,s}(t)$ model the bit rate constraints and they can be conceived as buffers of the amount of bit rate not currently satisfied. In the analysis of Section V, we showed that the objective of the proposed optimization solution is to stabilize the virtual queues so that the constraints are satisfied. Hence, if we continuously monitor the virtual queue lengths to detect when they are not stable, we can infer that the guarantees are not necessarily being satisfied. Therefore, our solution consists of adding a mechanism that monitors the evolution of virtual queues, so in the case when it detects a situation of a constant increment on the size of any of the virtual queues, it triggers an event to take the appropriate action.

It is important to note that this solution is very conservative, and the speed for detecting an isolation violation will depend on how the virtual queues' instability is measured. A *stabilization period* must also be considered at initialization, and every time a client connects or disconnects from the AP. This is because while the algorithm is finding stability, the size of the virtual queues may increase and stabilize at a larger length, and the scheduler must not consider this case as an unfeasible situation.

2) Enforcing Isolation: For the second stage, our proposal to solve the isolation issue and to obtain a feasible problem is to disconnect some clients from the AP (or to degrade its performance), in a controlled manner. The choice of which clients to disconnect may depend on several factors, such as the amount of consumed resources, the slice to which the client belongs, or the associated revenue. In a scenario with several APs and where a global *slicing architecture* is deployed, this decision should be performed by a high-level manager who may have a complete view of the available resources. Even more, with this approach, this manager may move the client to a different AP or network with more resources available so as to continue serving it. However, this decision problem is complex and is out of the scope of this work. Hereafter, we introduce a simple strategy for the actions to be taken when an isolation violation is detected.

For our solution, we propose to select a client and to remove its QoS guarantees, downgrading it to a *best-effort* client. With this approach, we allow the scheduler to assign resources to the client, but only if they are available. We propose selecting the client with the highest use of resources to be downgraded. If this still does not solve the isolation situation, we continue downgrading clients until it is resolved. Each downgrading is implemented by setting the bit rate guarantee K of the selected client to 0. As mentioned, the use of this policy is arbitrary, and other options are perfectly suitable.

VII. Implementation Details

In this section, we present brief descriptions about particular details that may need to be considered when implementing the proposed solution. Some of these details have already been implemented in our simulated prototype we analyze in Section VIII, but others are specific for an implementation in hardware.

A. Variable Time Slots

As was described in the system model, the proposed QoS Slicing solution is built over the allocation mechanism of the ATERR algorithm. For this, the ATERR is used with a fixed size quantum for all queues and with no adaptation mechanism. Hence, we can have a scheduler which assumes slotted time and on each slot assigns a client for transmission. However, because of how ATERR works, each time slot can be of variable size. Although the ATERR mechanism compensates these variations in the long run, it should be taken into account on the queue updates of each slot. Therefore, the proposed scheduler implementation receives feedback from the transmission module with the actual time slot size used. This feedback allows using the exact airtime consumed and not a fixed time slot for the updates of the (virtual and real) queues. Even more, variations may also happen because of the lack of data to fill an entire time slot. In those cases, when a queue empties before completing a time slot, the scheduler is executed to find a new queue for transmission.

B. Channel Capacity Estimation

As we already mentioned, an important aspect that influences the performance of the proposed solution is the ability to obtain a good estimation of the channel capacity of each client. In this regard, in the following, we explain two complementary approaches that can be taken.

On the one hand, most WiFi devices implement a *Modulation and Coding Scheme* (MCS) adaptation mechanism to find the most appropriate MCS for transmission given the current channel conditions. Also, for each MCS we have a transmission bit rate that can be achieved with it. Then, from this MCS adaptation mechanism, it is easy to obtain the current transmission bit rate used by the AP. Nowadays, the most widespread mechanism in WiFi is called *Minstrel* [33], which uses the frame loss rate to estimate the channel capacity. This mechanism, although not being optimal, provides a good estimation and has the advantage that can be obtained from real data, with almost no overhead.

However, the above approach provides just an upper bound on the actual channel capacity obtained. As was previously discussed, congestion at the wireless channel as well as retransmission because of collisions or packet losses will reduce the channel capacity finally obtained. Hence, to improve the channel capacity estimate, we also use the airtime consumed by the transmissions (including all the waiting times due to a busy medium, the back-off procedure, and the retransmissions) and the actual data that was transmitted. Therefore, we can estimate the capacity of the next slot by measuring the amount of data transmitted and the airtime consumed in the previous slot.

C. Maximum Arrival Rate

As discussed in previous sections, the proposed mechanism needs to know the maximum arrival rate per slot, which cannot always be easy to predict. One possibility is to always overestimate this value, which would guarantee the requested delay bound. However, a bad estimate would negatively impact on the obtained throughput, as more packets than needed would be dropped. The approach envisioned in this work is to request the tenant to inform the traffic characteristics of the slices as part of the agreement, for example, in the form of *average bit rate*, *peak bit rate*, and *burst size*.

A complementary strategy, which may also be necessary, is to add a traffic control mechanism to guarantee that the pre-defined contract is respected. This approach would guarantee that the delay is assured, but when the contract is not respected, it would shift the problem to the upper layers, where the packets may also be enqueued. Even more, this mechanism should be carefully designed to allow non-conforming traffic to ingress the system if enough free resources are available.

D. Parameter Calculation and Minimum Delay Bound

The proposed mechanism has two very important parameters that condition its performance and provide a trade-off between delay and throughput: V and $\epsilon_{n,s}$. As mentioned before, V determines the distance to the optimal utility; in our case, this means that as V is reduced the total throughput obtained by the AP increases. On the other hand, the delay bound linearly grows with it. In our supplementary material [29], we demonstrate that the upper bound on the packet delay is given by:

$$W_{n,s}^{max} = \left\lceil \frac{2V\omega + 3A_{n,s}^{max} + \epsilon_{n,s}}{\epsilon_{n,s}} \right\rceil \tag{39}$$

Hence, the delay does not also depends on V but also inversely depends on $\epsilon_{n,s}$, for each client and slice. Then, it is more appropriate to express the delay bound as a function of $V/\epsilon_{n,s}$.

TABLE I
SLICE AND TRAFFIC CONFIGURATION.

	Slice 1	**Slice 2**	**Slice 3**
Application	Live Video	Real-Time Gaming	Bulk
Traffic pattern	Constant Bit Rate of 300 kbps	Poisson Process[5] with 3 Mbps of mean rate.	TCP Bulk
Bit rate	300 kbps	3 Mbps	none
Delay	50 ms	25 ms	none
Capacity limit	30%	60%	10%
No. of flows	6	3	3

TABLE II
CHANNEL CAPACITIES

Client	**Slice**	**Average Capacity (Mbps)**	
		Scenario 1	**Scenario 2**
1	1	5	5
2	1	5	5
3	1	10	6
4	1	10	8
5	1	20	13
6	1	20	15
7	2	20	15
8	2	30	22
9	2	30	22
10	3	30	30
11	3	10	10
12	3	5	5

In our QoS Slicing context, it is important to determine, for a particular scenario and configuration, the minimum delay bound a system can provide. Let us observe that for a given V parameter, the minimum possible delay bound is given by the maximum possible value of $\epsilon_{n,s}$. From previous assumptions we know that $\epsilon_{n,s} \leq A_{n,s}^{max}$. Hence, each client's minimum delay bound is given by:

$$W_{n,s}^{max} = \left\lceil \frac{2V\omega + 4A_{n,s}^{max}}{A_{n,s}^{max}} \right\rceil \tag{40}$$

Note that the previous result provides a bound on the delay measured in time slots. Hence, the actual delay measured in seconds will depend on the particular time slot selected. In our case, two parameters will have an important impact on the time slot value, the *quantum* of the ATERR mechanism, and the maximum packet transmission time, which, as discussed in [7], affects the deviation from the predefined quantum. This last factor is really important since a very low channel capacity in some clients can generate long delays. For example, if the wireless medium allows a channel capacity of 2 Mbps, a packet of 1500 Bytes would take 6ms to be transmitted, limiting the achievable queue delay to a value higher than that.

Therefore, the minimum channel capacity allowed in the wireless device should be controlled, and clients with very low capacity might need to be dropped or moved to another device. Note that a client with low capacity would affect all clients.

VIII. EVALUATION

In this section, we evaluate the behavior and performance of the proposed slicing mechanism by implementing it on the MATLAB Simulink [34] software. In the prototype, we model the queue and scheduling operation, the input traffic patterns, as well as the variable channel conditions of the wireless links. The implemented model is available at [35]. The goal of the evaluation is to show how our solution provides the QoS guarantees to slices with different requirements when deployed on a WiFi network. We tested slices with different traffic patterns and different QoS requirements, having clients with different and variable channel conditions.

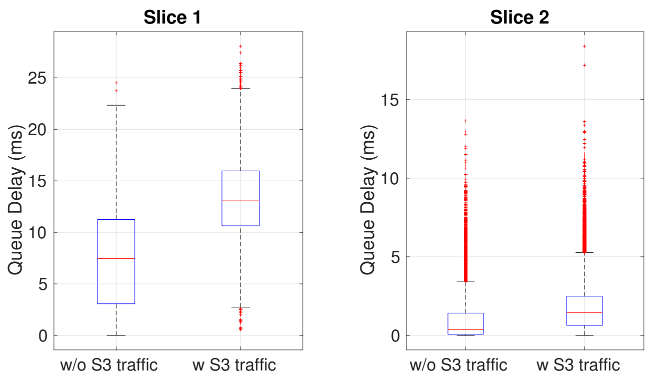
[5]We choose to model the traffic as a Poisson Process to allow variability on the offered load. However, actual gaming traffic may follow a different pattern.

A. Simulation Setup

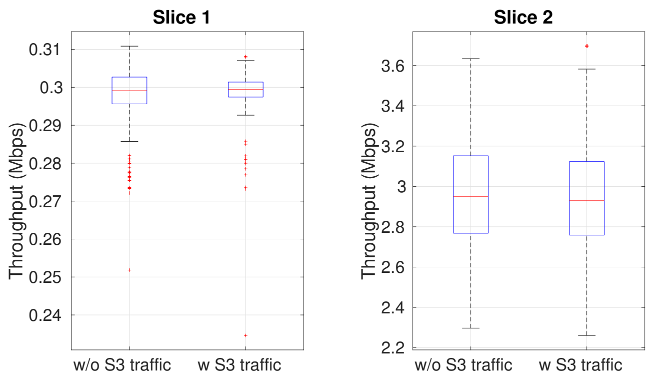
The simulation scenario is composed of a single WiFi Access Point (AP) and several clients that connect to such AP. We consider three slices deployed at the AP: Slices 1, 2 and 3, for Video Live Streaming (Video Conference), Real-Time Gaming, and Bulk Background traffic, respectively. The scenario comprises 12 clients, which are enough to assess the behavior of the proposed scheme, but also facilitate the analysis of the results. Each client belongs to just one slice and receives traffic from one application, thus resulting in 12 traffic flows in our system. For each flow, there is a traffic generator, which sends packets (following a given pattern) to be delivered to the clients from the AP.

The QoS requests of each slice and the traffic patterns of each flow are summarized in Table I. Slice 1 requires a minimum guaranteed bit rate of 300Kbps for each flow, with a maximum allowed delay (delay bound) of 50ms. This must be respected while the required resources do not exceed the 30% of the total resources of the AP. Slice 2 requests a guaranteed bit rate of 3Mbps, with a maximum delay of 25ms, and with a resource limit of 60%. For both slices, the average bit rate guarantee must be measured in a time window of 1 second and includes a tolerance of a 10% from the requested average. Finally, Slice 3 only requests a maximum of 10% of the AP resources, with no QoS guarantees. This slice will bear TCP flows with varying loads. The TCP traffic generator consists of an application that sends data as fast as possible, as it would be the case of a bulk file transfer. In addition, when the lower layer buffer is full, it waits until some frames are dequeued to send more data, as it would be the case of a real elastic service.

The 802.11 MAC layer behavior is mimicked with a simple model that estimates the overall time of a packet transmission (from being available at the transmitter until it is correctly received at the destination) for a certain channel occupancy and probability of collision. The model considers the most important aspects of the 802.11 Access Control mechanism: the time a packet is waiting for the medium to be free, its transmission time at the wireless interface, and the waiting time due to contention. The model receives an average channel capacity C, which can be different for each client connected to the AP, depending on the distance between the client and the AP and on the channel occupancy. A collision probability p can also be configured, and the 802.11 exponential backoff algorithm is used to establish the contention window size for consecutive retransmissions of a single frame, after successive collisions. Then, the overall time duration of a packet is finally computed, based on the aforementioned inputs and on the packet length. In the simulations shown in this work, the model is configured with the average capacities depicted in Table II, with a probability $p = 0.1$ and all the necessary constant values (CW_{min}, slot size) from the 802.11n standard. Note that more dense scenarios (i.e., having a higher channel occupancy) could be easily captured by decreasing the corresponding capacities or increasing the collision probability.

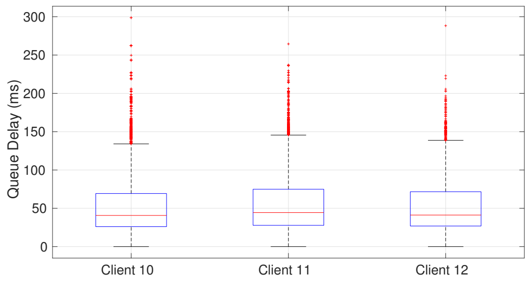


(a) Worst Client Delay

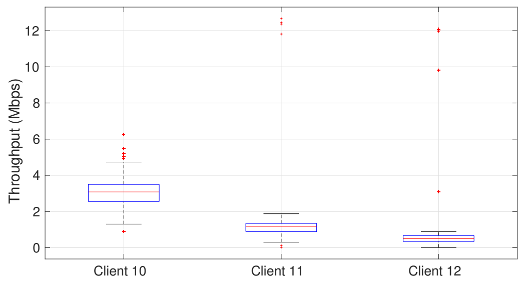


(b) Worst Client Throughput

Fig. 1. Scenario 1. Delay and Throughput for Slices 1 and 2 with and without Traffic on Slice 3.



(a) Delay



(b) Throughput

Fig. 2. Scenario 1. Delay and Throughput for Slice 3.

For the system configuration, we have chosen a quantum size of $q = 2.5ms$ for all slices and clients. Under this configuration, given the airtime allocation mechanism used, the maximum possible airtime consumption of a client on a round is $4.8ms$ (see [7]). Then, we select the values $V = 3$, $\epsilon_1 = 1$ for all clients of Slice 1, and $\epsilon_2 = 4.55$ for all clients of Slice 2. Using the delay bound (39), the selected parameter values yield an upper delay bound for Slice 1 of $48ms$ and for Slice 2 of $30ms$.

Furthermore, the system setup also includes the CoDel [13] queue management technique on the queues associated with Slice 3. Although CoDel does not provide a bound on the delay, it helps in controlling the queue lengths and on minimizing the delay.

We evaluate three different resource usage scenarios (different offered loads):

- **Scenario 1 (Loose Resource Usage)** Given the clients' capacities and the offered load, all QoS guarantees can be accomplished without using the 100% of the requested resources.
- **Scenario 2 (Tight Resource Usage)** Similar to the previous scenario, but the slices use almost all the requested resources.
- **Scenario 3 (Lack of Resources)** After an initial resource assignment, because of a variation in the channel capacities, the amount of available resources is not sufficient to comply with all QoS requirements.

B. *Results for Scenario 1*

For the first scenario, Loose Resource Usage, we show results with and without traffic in Slice 3. The objective is to assess how our solution correctly manages isolation and guarantees QoS requirements from Slice 1 and Slice 2. Because of space constraints and to facilitate visualization, we are not able to show the performance of all 12 clients, but we show, for each slice, the highest obtained delay and the lowest throughput (the worst client). The channel capacities of each client are depicted in Table II. Clients 1 to 6 belong to Slice 1, Clients 7 to 8 to Slice 2, and Clients 10 to 12 to Slice 3.

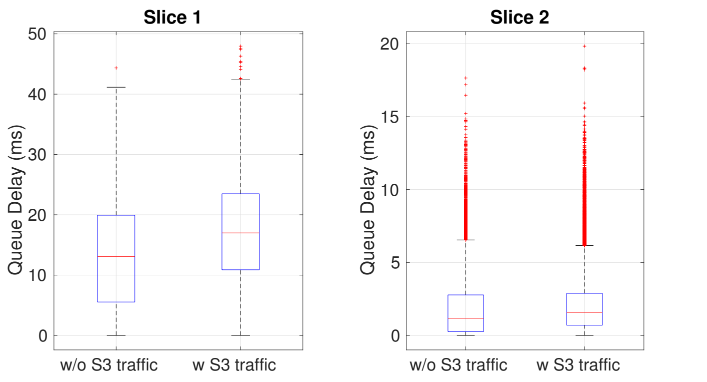
In Figure 1 is shown the obtained delay and throughput. It can be observed that in both cases (with and without traffic on Slice 3), the QoS requirements of delay and bit rate are always guaranteed. When traffic is generated in Slice 3, there are some variations on the average delay, but the maximum delay is kept below the required bound.

For the experiment, we calculate the packet drop ratio for the clients of Slices 1 and 2, which result in a ratio below 10^{-2} for all clients. We also compute the percentages of airtime allocated to the different slices, obtaining the following median values: Slice 1 - 23.6%; Slice 2 - 39.7%; Slice 3 - 36.4%. Given that Slices 1 and 2 require fewer resources than the maximum capacity to fulfill the QoS requirements, the remaining resources are then exploited by Slice 3.

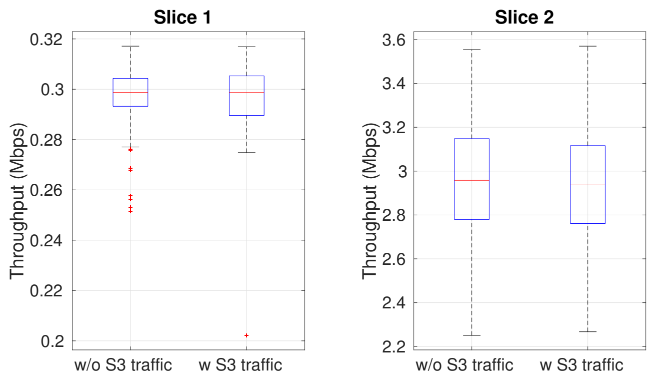
Figure 2 depicts the obtained performance by the clients of Slice 3. It shows how extra available resources are used by Slice 3 to obtain throughputs between 0.5 and 3 Mbps for its clients. As previously shown, this is achieved without affecting the performance of the other slices.

C. *Results for Scenario 2*

As previously explained, we also analyzed a scenario where Slices 1 and 2 require more resources to fulfill the QoS requirements. The average client capacities are also depicted in Table II.

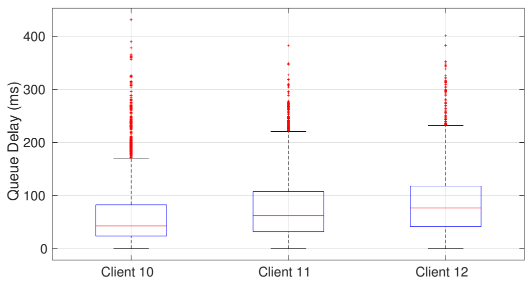


(a) Worst Client Delay

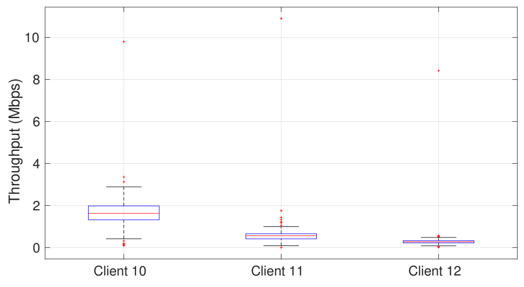


(b) Worst Client Throughput

Fig. 3. Scenario 2. Delay and Throughput for Slices 1 and 2 with and without Traffic on Slice 3.



(a) Delay



(b) Throughput

Fig. 4. Scenario 2. Delay and Throughput for Slice 3.

Figures 3 and 4 show the results that were obtained for this scenario. As in the previous case, it can be observed that all QoS requirements are correctly met. However, we can notice a slightly worse overall system performance. This is because the total channel capacity is lower than before, and more resources are needed to fulfill the required performance.

In this case, the obtained airtime allocation is: Slice 1 - 27.9%; Slice 2 - 52.9%; Slice 3 - 18.3%. This result shows that the proposed solution adapts the resource usage, and it limits Slice 3 traffic, to provide the required QoS in Slices 1 and 2. This can also be observed in Figure 4 where the obtained performance of Slice 3 is reduced.

D. Results Scenario 3

Finally, we also evaluate a scenario where, in a given instant, the channel capacities deteriorate up to a point such that there are not enough resources to comply with all the QoS requirements. We start the simulation with the same channel capacities used in Scenario 2, and after 50 seconds, we move Client 9 further away from the AP, so that the average channel capacity decreases to 5 Mbps. It is worth noting that this scenario generates a case of isolation violation, as there is not any possible allocation of resources that can accomplish the required guarantees (there is no feasible solution to the optimization problem). Hence, our goal with this experiment is to show two aspects of our isolation solution:

1) A variation for a client belonging to one slice does not affect the QoS guarantees of the other slices.
2) When a slice cannot assure the required QoS guarantees, the problem is detected, and an appropriate action is taken.

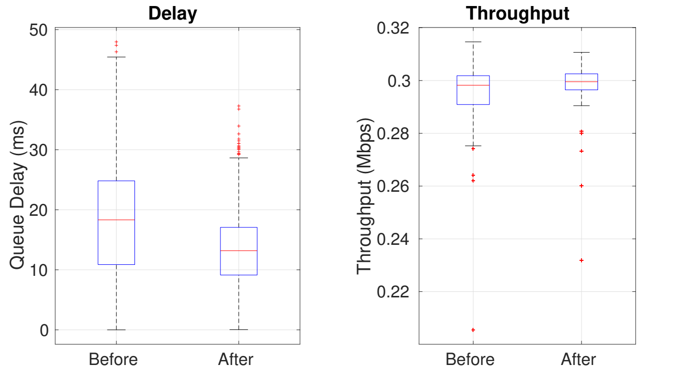
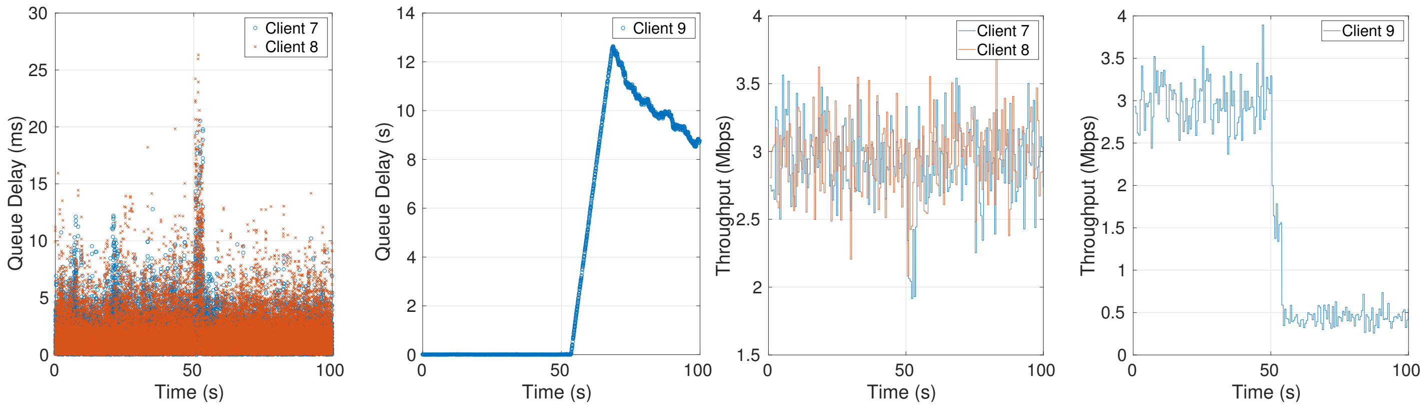


Fig. 5. Scenario 3. Delay and Throughput for Slice 1 before and after Change in Slice 2.

For the first aspect, we provide in Figure 5 the performance results of Slice 1, before and after the channel variation of Slice 2. As expected, the variation does not affect the QoS requirements of Slice 1, guaranteeing the isolation between slices. Secondly, in Figure 6, we depict the evolution along the simulation time of the achieved throughput and delay of the three clients of Slice 2. In this case, as explained in Section VI, the scheduler detects the problem by monitoring the virtual queues, selects Client 9, and removes it from Slice 2. Hence, as can be seen, after a small transient period (4 seconds), in which all the clients of the slice are affected by the drastic change in the channel capacity of Client 9, the mechanism stops considering Client 9 QoS requirements. Then, the other clients recover their previous performance, while Client 9 is swapped to a best-effort client.

IX. Conclusions

Slicing has become an essential part of the current 5G networks design. In this context, providing WiFi networks with



(a) Delay (b) Throughput

Fig. 6. Scenario 3. Delay and Throughput Evolution for Slice 2.

the ability to implement slicing further facilitates the integration of this technology in the 5G ecosystem. In this paper, we proposed a dynamic resource allocation mechanism to support the development of network slicing in WiFi Access Points. Through a novel packet scheduling design, slices with diverse QoS requirements can be defined.

The QoS Slicing problem is formulated as a stochastic optimization problem, which maximizes the total average throughput of the AP while satisfying the constraints of minimum average bit rate, bounded delay, and average airtime usage limit. We consider this is a relevant contribution, as the problem of resource allocation to implement QoS Slicing in wireless networks (with all its complexities and particularities) is condensed on a single optimization problem. The theory of Lyapunov Optimization was applied to transform the stochastic optimization problem into a deterministic problem that must be solved on each time slot. The obtained solution just requires the instantaneous value of the channel capacity and the queues' status and consists of solving a deterministic optimization problem on each time slot. From the obtained solution, it was derived a scheduling algorithm that selects, in each time slot t, the client to transmit and the packets to be dropped.

Nevertheless, to implement the obtained solution in WiFi APs, we employ the ATERR airtime-allocation scheduler to provide an approximate time-slotted system. Even more, thanks to the ATERR scheduling strategy and the system model developed, the obtained scheduling algorithm consists of finding a maximum on each time slot. This is important since it makes the complexity of the algorithm to be linear on the number of clients, and assures the scalability of the solution. Lastly, we contribute with a mechanism to detect and control unfeasible situations, where more resources than needed are necessary to achieve the requested guarantees. This provides isolation guarantees to slices in scenarios where unexpected channel capacity variations appear.

Finally, we carried out an extensive simulation-based analysis of the proposed scheduler, evaluating the performance in a typical slicing scenario. The results show the effectiveness of our solution in guaranteeing the QoS requirements of all slices, while also providing the required isolation between slices. In summary, in this work, we contributed with a novel mechanism to implement QoS Slicing in WiFi APs, which guarantees a requested minimum average bit rate with bounded delay to each client of a slice. Although the mechanism was designed employing an existing technique, its application to a complex and concrete problem in the context of WiFi technology brought many new challenges that were successfully tackled.

In the future, we plan to implement our solution in a network simulator to evaluate more complex scenarios. This includes the interaction of the proposed solution with a high-level network manager which controls slices and clients connections. We will use such scenarios to assess the performance of the proposed scheduler in terms of packet drops. Moreover, we are working to deploy the proposed scheme over real platforms, both to assess its feasibility and to broaden its performance analysis, realistically considering the particularities of WiFi technology. Such real deployment would also allow to study the scalabilty of the proposed approach, regarding the required number of queues in the system, as well as the execution complexity of the algorithm. Finally, we also plan to further investigate on the applied theory to remove some of the needed assumptions and to theoretically analyze the algorithm behavior in more detail. In particular, we are interested in obtaining a bound for the average bit rate in a slot, and in assessing how fast the performance converges to the required value.

Acknowledgment

This work has been supported in part by the European Commission and the Spanish Government (Fondo Europeo de Desarrollo Regional, FEDER) by means of the EU H2020 NECOS (777067) and ADVICE (TEC2015-71329) projects, respectively.

References

[1] M. Richart, J. Baliosian, J. Serrat, and J.-L. Gorricho, "Resource slicing in virtual wireless networks: A survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 462–476, 2016.

[2] 3GPP, "Service requirements for the 5G system; Stage 1 (Release 15)," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 22.261, 12 2018, version 15.7.0.

[3] ——, "NR; NR and NG-RAN Overall Description; Stage 2 (Release 15)," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.300, 04 2019, version 15.5.0.

[4] K. Katsalis, K. Choumas, T. Korakis, and L. Tassiulas, "Virtual 802.11 wireless networks with guaranteed throughout sharing," in *Computers and Communication (ISCC), 2015 IEEE Symposium on*. IEEE, 2015, pp. 845–850.

[5] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[6] M. Richart, J. Baliosian, J. Serrati, J.-L. Gorricho, R. Agüero, and N. Agoulmine, "Resource allocation for network slicing in wifi access points," in *Network and Service Management (CNSM), 2017 13th International Conference on*. IEEE, 2017, pp. 1–4.

[7] M. Richart, J. Baliosian, J. Serrat, J.-L. Gorricho, and R. Agüero, "Slicing in wifi networks through airtime-based resource allocation," *Journal of Network and Systems Management*, vol. 27, no. 3, pp. 784–814, Jul 2019.

[8] ——, "Guaranteed bit rate slicing in wifi networks," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, IEEE, Ed., April 2019, pp. 1–8.

[9] A. Malik, J. Qadir, B. Ahmad, K.-L. A. Yau, and U. Ullah, "Qos in ieee 802.11-based wireless networks: a contemporary review," *Journal of Network and Computer Applications*, vol. 55, pp. 24–46, 2015.

[10] A. Banchs and X. Perez, "Providing throughput guarantees in ieee 802.11 wireless lan," in *2002 IEEE Wireless Communications and Networking Conference Record. WCNC 2002*, vol. 1. IEEE, 2002, pp. 130–138.

[11] A. Banchs, P. Serrano, and L. Vollero, "Providing service guarantees in 802.11 e edca wlans with legacy stations," *IEEE Transactions on Mobile Computing*, vol. 9, no. 8, pp. 1057–1071, 2010.

[12] J. Gettys, "Bufferbloat: Dark buffers in the internet," *IEEE Internet Computing*, no. 3, p. 96, 2011.

[13] K. Nichols and V. Jacobson, "Controlling queue delay," *Communications of the ACM*, vol. 55, no. 7, pp. 42–50, 2012.

[14] T. Høiland-Jørgensen, P. Hurtig, and A. Brunstrom, "The good, the bad and the wifi: Modern aqms in a residential setting," *Computer Networks*, vol. 89, pp. 90–106, 2015.

[15] T. Høiland-Jørgensen, M. Kazior, D. Täht, P. Hurtig, and A. Brunstrom, "Ending the anomaly: Achieving low latency and airtime fairness in wifi," in *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. Santa Clara, CA: USENIX Association, 2017, pp. 139–151.

[16] A. Asadi and V. Mancuso, "A survey on opportunistic scheduling in wireless communications," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1671–1688, 2013.

[17] M. Andrews, K. Kumaran, K. Ramanan, A. Stolyar, P. Whiting, and R. Vijayakumar, "Providing quality of service over a shared wireless link," *IEEE Communications magazine*, vol. 39, no. 2, pp. 150–154, 2001.

[18] J.-W. Lee, R. R. Mazumdar, and N. B. Shroff, "Opportunistic power scheduling for multi-server wireless systems with minimum performance constraints," in *IEEE INFOCOM 2004*, vol. 2. IEEE, 2004, pp. 1067–1077.

[19] B. Sadiq, S. J. Baek, and G. De Veciana, "Delay-optimal opportunistic scheduling and approximations: The log rule," *IEEE/ACM Transactions on Networking (TON)*, vol. 19, no. 2, pp. 405–418, 2011.

[20] H. Kim and G. De Veciana, "Losing opportunism: Evaluating service integration in an opportunistic wireless system," in *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*. IEEE, 2007, pp. 982–990.

[21] M. J. Neely, "Opportunistic scheduling with worst case delay guarantees in single and multi-hop networks," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 1728–1736.

[22] M. Carmo, S. Jardim, A. Neto, R. Aguiar, D. Corujo, and J. J. Rodrigues, "Slicing wifi wlan-sharing access infrastructures to enhance ultra-dense 5g networking," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[23] E. Coronado, R. Riggio, J. Villa1ón, and A. Garrido, "Lasagna: Programming abstractions for end-to-end slicing in software-defined wlans," in *2018 IEEE 19th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2018, pp. 14–15.

[24] G. Bhanage, D. Vete, I. Seskar, and D. Raychaudhuri, "Splitap: leveraging wireless network virtualization for flexible sharing of wlans," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. IEEE, 2010, pp. 1–6.

[25] K. Nakauchi, Y. Shoji, and N. Nishinaga, "Airtime-based resource control in wireless lans for wireless network virtualization," in *Ubiquitous and Future Networks (ICUFN), 2012 Fourth International Conference on*. IEEE, 2012, pp. 166–169.

[26] M. Derakhshani, X. Wang, T. Le-Ngoc, and A. Leon-Garcia, "Virtualization of multi-cell 802.11 networks: Association and airtime control," *arXiv preprint arXiv:1508.03554*, 2015.

[27] K. Guo, S. Sanadhya, and T. Woo, "Vifi: virtualizing wlan using commodity hardware," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 18, no. 3, pp. 41–48, 2015.

[28] K. Koutlia, A. Umbert, R. Riggio, I. Vilà, and F. Casadevall, "A new ran slicing strategy for multi-tenancy support in a wlan scenario," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 2018, pp. 64–70.

[29] M. Richart, J. Baliosian, J. Serrat, J.-L. Gorricho, and R. Agüero, "Supplementary material for slicing with guaranteed quality of service in wifi networks," June 2020, supplementary material not published. [Online]. Available: http://doi.org/10.5281/zenodo.3908729

[30] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[31] L. Georgiadis, M. J. Neely, and L. Tassiulas, *Resource allocation and cross-layer control in wireless networks*. Now Publishers Inc, 2006.

[32] A. Saeed, N. Dukkipati, V. Valancius, C. Contavalli, A. Vahdat *et al.*, "Carousel: Scalable traffic shaping at end hosts," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 404–417.

[33] J. Berg, *Minstrel Rate Control Algorithm Documentation*, 2016. [Online]. Available: https://wireless.wiki.kernel.org/en/developers/documentation/mac80211/ratecontrol/minstrel

[34] I. The MathWorks, *MATLAB Simulink version 9.2 (R2018b)*, The MathWorks, Inc., Natick, Massachusetts, United States, 2018.

[35] M. Richart, "Qos slicing simulation code for matlab simulink (version 0.2)," June 2020. [Online]. Available: http://doi.org/10.5281/zenodo.3898849

Matías Richart received his Computer Engineer degree from the University of the Republic (Uruguay) in 2011 and his Ph.D. from University of the Republic and from Polytechnic University of Catalonia (Spain) in 2019. His research interests include autonomic control and resource management of wireless networks, virtual wireless networks, network slicing, and network simulation. Currently, he is with the Department of Computer Science at the University of the Republic.

Javier Baliosian received the degree of Computer Engineer from the University of the Republic (Uruguay) in 1998, and his Ph.D. from Polytechnic University of Catalonia (Spain) in 2005. Since then has been involved in several research projects with different groups such as the Computer Laboratory of the University of Cambridge, the Laboratory of Communication Networks at KTH, and the Ericsson Ireland Research Centre, where he worked as a researcher and project coordinator until late 2007. Currently, he is with the Department of Computer Science at the University of the Republic and the Universitat Politècnica de Catalunya (UPC).

Joan Serrat received a degree of Telecommunications Engineering in 1977 and a Ph.D. in the same field in 1983, both from the Universitat Politècnica de Catalunya (UPC). Currently, he is a Full Professor at UPC, where he has been involved in several collaborative projects with different European research groups, both through bilateral agreements or through participation in European funded projects. His topics of interest are in the field of autonomic networking and service and network management. Currently, he is the contact point of the TM Forum at UPC.

Juan-Luis Gorricho received a telecommunication engineering degree in 1993, and a Ph.D. degree in 1998, both from the UPC. He is currently an associate professor at the UPC. His recent research interests are in applying artificial intelligence to ubiquitous computing and network management; with a special interest in using smartphones to achieve the recognition of user activities and locations; and applying linear programming and reinforcement learning to resource management in virtualized networks and functions.

Ramón Agüero is an Associate Professor at the University of Cantabria. He received his MSc in Telecommunications Engineering (1st class honors) in 2001 and the Ph.D. (hons.) in 2008. His research focuses on future network architectures, especially regarding the (wireless) access part of the network and its management. He is also interested in Network Coding. He serves in the Editorial Board of IEEE Communication Letters (Senior Editor), IEEE Open Journal of the Communications Society, and Wireless Networks (Springer). He is a senior member of IEEE since 2015.