Guaranteed Bit Rate Slicing 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*—In forthcoming 5G networks, slicing has been proposed as a means to partition a shared physical network infrastructure into different self-contained logical parts (slices), which are set up to satisfy certain requirements. Although the topic has been thoroughly investigated by the scientific community and the industry, there are not many works addressing the challenges that appear when trying to exploit slicing techniques over WiFi networks. In this paper, we propose a novel method of allocating resources for WiFi networks to satisfy minimum bit rate requirements. We formulate an optimization problem, and we propose a solution based on the theory of Lyapunov drift optimization. The validity of the proposed solution is assessed by means of a simulation-based evaluation in Matlab.**

I. Introduction

Network slicing has recently been proposed as one of the main enablers for 5G networks. Using this paradigm, infrastructure owners are able to 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. Most existing works on slicing in wireless networks consider slicing mostly for cellular networks and, in particular, for LTE technology. In this paper we instead focus on the IEEE 802.11 (WiFi) technology, for which slicing has not been thoroughly studied, in spite of its doubtless relevance. We analyze the case of *Quality of Service Slicing*, and we propose a mechanism to achieve guaranteed bit rate slices in WiFi Access Points. More precisely, we develop a mechanism for implementing guaranteed bit rate slices in a WiFi Access Point (AP) by considering the transmission time (airtime) as the resource to share. We propose a traffic queuing and scheduling technique to allocate the AP transmission time, satisfying the bit rate requirements. The resource allocation is indeed efficient, as each slice receives the exact resources that were guaranteed, and free resources can be thus used by additional slices.

We formulate the guaranteed bit rate slicing problem as a stochastic optimization problem, where the channel conditions and the arrival rates are unknown stochastic processes. Our solution is constructed following the *Stochastic Network Optimization* framework described in [1] and [2], which is based on Lyapunov optimization theory. This framework permits to obtain an equivalent deterministic problem, which provides an approximate bounded solution to the original one.

Although network slicing is a rather new concept, the problem of providing Quality of Service in IEEE 802.11 networks has been thoroughly studied in the last twenty years. However, the vast majority of the research on this subject has focused on providing QoS to the transmissions from the stations to the AP, or between stations [3]. This is due to the fact that WiFi networks have traditionally been 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 to stations, the works of Banchs et. al. [4], [5] are worth mentioning. In those proposals throughput is guaranteed through access management schemes, controlling the Contention Window (CW) size. On the other hand, in our proposal we only consider the 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 slicing, there are few works dealing with this issue in WiFi devices. In [6], [7], [8], [9], [10] slicing is achieved through the allocation of airtime ratios to each slice, which does not guarantee any type of QoS. The authors of [11] 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 does not guarantee a minimum bit rate to the slices. An extensive review on recent proposals for slicing in WiFi can be found in our previous paper [12]. The main differences of the proposal discussed herewith from those previous works are that it does not modify or control low-level MAC parameters, neither it needs feedback from the medium or the stations to achieve the required allocation. It only needs information on the airtime consumed and from the local rate manager, which can be obtained from the hardware driver. In addition, we avoid traffic shaping, which can lead to inefficiencies if not controlled adequately. Furthermore, we use the queuing model proposed in our previous work [13], which contemplates the particularities of the hardware behaviour so as to avoid queue buildup at lower layers and to allow packet aggregation.

The rest of the paper is organized as follows: in §II we introduce our vision of network slicing, and we define the concept of *Quality of Service Slicing*; in §III we present our system model and problem formulation; in §IV and §V we discuss the solution to the optimization problem and we depict the proposed scheduling mechanism. In §VI we show some experimental evaluation of our proposal and, finally, in §VII we conclude the paper, identifying some aspects that will be tackled in our future research.

II. Network Slicing

Although different definitions of *network slicing* have been proposed, in this paper we use the definition introduced in our previous work [12]. As already mentioned, a slice is a partition of network resources that are allocated by a *infrastructure provider* to a *tenant* for its particular use. On the other hand, from a data level perspective, a slice can be considered as a set of traffic flows with some common features, demanding some performance requirements. A slice can support flows from multiple *clients* (mobile clients of the network in our case), but at the same time, a client can participate in multiple slices. However, a flow belongs to a single slice, and slices are always independent between each other. Hence, two main variants of slicing can be considered: *Quality of Service Slicing* (QoSS), and *Infrastructure Sharing Slicing (ISS)* [12]. In this work we focus on QoSS.

A. QoS Slicing

In this slicing variant tenants require slices with some Quality of Service performance objectives, which should be provided by the network infrastructure. Those are diverse but some common choices are: packet delay, jitter, packet loss ratio, and bit rate.

Regardless of the QoS parameter considered, some common design choices should be taken into account. The first decision to make is whether the performance requirements specified for a slice are individually applicable to every flow that belongs to such slice or otherwise to the aggregated traffic of the slice. While there are other approaches in the literature, where performance guarantees are requested for the entire slice and not for each flow [11], in this paper we consider the QoS requirements of a slice as the QoS to be guaranteed to every flow within the slice. Another important design aspect of the QoS variant is the flexibility of the QoS guarantees. In wireless communications, and particularly in WiFi, it is necessary to allow a certain tolerance for the QoS requirements, and a policy that defines the actions to take when it is not possible to meet the requirements. There are two main reasons for this: first, in wireless communications it may happen that the capacity of an client's channel is not enough to support a particular requirement; second, in unlicensed spectrum technologies like WiFi, the wireless medium could be saturated by other communications or by any type of interference, which are out of the control of the network. A tolerance on the required QoS guarantee would thus add flexibility, allowing the system to adapt to environment changes. For example, a guaranteed bit rate requirement may include a tolerance on the percentage of time it is not accomplished. Moreover, a policy is needed for when the guarantees, even with a tolerance, are not fulfilled and some action should be taken. For example, a policy may specify that when the guarantee is not met, the corresponding client connection should be dropped, notifying the service provider.

Finally, a key aspect of slicing is to ensure slices do not disturb each other, avoiding performance degradation or appropriation of resources allocated to other slices. This is called *isolation*, and it must be considered in the implementation of QoS Slicing, as clients with low channel capacity can indeed jeopardize the performance of the whole network.

III. QoS Slicing Model and Problem Formulation

As already mentioned, our focus is on the network's edge, more precisely, on WiFi APs. The objective is to implement *QoS Slicing* by allocating the necessary resources to the different slices. In particular, we describe the problem of guaranteeing a minimum bit rate to each client of a slice, in the context of a *WiFi* AP. In the following, we define our model and we formulate the resource allocation problem on an AP as an optimization problem.

A. Slicing Model

In our approach we consider slotted time (in Section IV we discuss how the time can be slotted for WiFi). Lets consider the following parameters for a given AP:

- S is the number of slices defined in the AP, such that $s \in [1, S]$ identifies a slice.
- N_s is the number of clients in slice s where $n_s \in [1, N_s]$ identifies a client.
- Each slice has the requirement K_s of a minimum bit rate, which must be guaranteed to every client of the slice.
- C_{n_s} is the channel capacity between the AP and the client n_s. This capacity is variable and it depends on the channel conditions. For our formulation we will consider the capacity to be invariant within a time slot, hence, we define $C_{n_s}(t)$ as the capacity between the AP and client n_s in time slot t.

The guaranteed bit rate slicing problem consists on how the AP should schedule the transmissions to the different clients to guarantee that each client receives the minimum bit rate ensured by the corresponding slice.

Lets first observe that the bit rate obtained by a client depends mostly 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_s, in time slot t, is given by:

$$R_{n_s}(t) = C_{n_s}(t) \times x_{n_s}(t) \tag{1}$$

where $x_{n_s}(t)$ is the proportion (or ratio) of time slot t assigned for transmitting to client n_s.

The problem is to find the ratios x_{n_s} which guarantee that the slice requests are satisfied. However, we have not yet exactly defined the requests and how they can be satisfied. Note that there exist many ways of defining a minimum bit rate request: for example, the requirement can be very strong such as ensuring that the minimum bit rate is always satisfied (at every time slot), or it can be more relaxed, for instance requiring that the minimum bit rate is satisfied on average.

In this proposal, we consider the second option, where slice requests consist on 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}) to be above the requested average bit rate K_s.

Another important aspect to note is that $R_{n_s}(t)$ is a random (or stochastic) process, because the channel capacity can vary

randomly depending on several factors. Hence, for our problem formulation we will consider the expected time average of the bit rate, defined as:

$$\overline{R}_{n_s} = \lim_{T\to\infty} \frac{1}{T} \sum_{t\in T} \mathbb{E}\{R_{n_s}(t)\} \tag{2}$$

$$= \lim_{T\to\infty} \frac{1}{T} \sum_{t\in T} \mathbb{E}\{C_{n_s}(t) \cdot x_{n_s}(t)\} \tag{3}$$

In the following formulation, which just focuses on the allocation problem, it is assumed that the slices' 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 migrate slices when more resources than those available are needed. Also, in our approach we only consider downlink traffic, i.e. traffic from the AP to the clients.

B. Problem Formulation

Based on the above description, the problem is to find, for every slot t, the assignment vector, $\boldsymbol{x}(t) = \{x_{1_1}(t), x_{2_1}(t), ..., x_{N_1}(t), ..., x_{1_S}(t), ..., x_{N_S}(t)\}$, which guarantees that all slices' requests satisfy the following:

$$\overline{R}_{n_s} \geq K_s,\ \forall s \in [1, S],\ \forall n_s \in [1, N_s]. \tag{4}$$

If we add to the above problem the objective of maximizing the average bit rate of the entire AP, so as to use the resources efficiently, we can formulate a *stochastic optimization problem*:

$$\underset{\boldsymbol{x}}{\text{maximize}} \quad \sum_{s=1}^{S} \sum_{n_s=1}^{N_s} \overline{R}_{n_s} \tag{5}$$

$$\text{subject to} \quad \overline{R}_{n_s} \geq K_s,\ \forall s \in [1, S],\ \forall n_s \in [1, N_s], \tag{6}$$

$$\sum_{s=1}^{S} \sum_{n_s=1}^{N_s} x_{n_s}(t) \leq 1, \tag{7}$$

$$0 \leq x_{n_s}(t) \leq 1. \tag{8}$$

In this optimization problem the objective is to find the transmission ratios $x_{n_s}(t)$ to maximize the total average bit rate. Note that $\boldsymbol{x} = \{\boldsymbol{x}(1), ..., \boldsymbol{x}(T)\}$ is the vector of assignment vectors for all the clients at all slots.

Constraint (6) considers the minimum average bit rate K_s of each slice. Note that it represents in fact a set of constraints, where there is a different one for each client of each slice. Constraints (7) and (8) control that no more resources than those available are assigned, by limiting the possible values of the $\boldsymbol{x}$ variables.

The main complexity of the optimization problem (5)-(8) resides in the variability of the channel capacity $C(t)$. As already mentioned, $C(t)$ can be considered a stochastic process, for which we do not know its distribution and so, cannot be predicted.

C. Adding fairness

Any solution to the proposed previous problem can be improved for cases when there are more resources available than requested. Note that the problem formulation in (5)-(8) is based on the objective of total throughput maximization, which will lead to solutions where the client with the highest capacity hoards all the unused resources. To prevent this we propose to introduce fairness in the objective formulation.

A very common approach is to use *proportional fairness* [14] for the allocation. We define $\mathcal{U}$ as the the set of all achievable utilities and $u_i(\boldsymbol{x})$ an utility function, which returns the profit of a given allocation vector. When $\mathcal{U}$ is convex, a proportional fair allocation can be obtained as the solution to the following problem [14]:

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{maximize}} \quad & \sum_{i\in N} \log u_i(\boldsymbol{x}) \\ \text{subject to} \quad & u \in \mathcal{U} \end{aligned} \tag{9}$$

As our objective is to achieve fairness on the allocations of the remaining resources after guaranteeing the minimum bit rates, and not fairness on the global allocations, we define the utilities of each client as the difference between the allocated bit rate $R_{n_s}(t)$ and the minimum bit rate required K_s:

$$u_{n_s}(t) = R_{n_s}(t) - K_s, \tag{10}$$

and the fair utility function as:

$$\phi(\boldsymbol{u}(t)) = \sum_{s=1}^{S} \sum_{n_s=1}^{N_s} \log(u_{n_s}(t)). \tag{11}$$

Hence, we can reformulate our stochastic optimization problem in (5)-(8) so as to consider proportional fairness:

$$\underset{\boldsymbol{x}}{\text{maximize}} \quad \phi(\overline{\boldsymbol{u}}) \tag{12}$$

$$\text{subject to} \quad \overline{u}_{n_s} \geq 0\ \forall s \in [1, S],\ \forall n_s \in [1, N_s], \tag{13}$$

$$\sum_{s=1}^{S} \sum_{n_s=1}^{N_s} x_{n_s}(t) \leq 1, \tag{14}$$

$$0 \leq x_{n_s}(t) \leq 1. \tag{15}$$

where

$$\phi(\overline{\boldsymbol{u}}) = \sum_{s=1}^{S} \sum_{n_s=1}^{N_s} \log(\overline{u}_{n_s}) \tag{16}$$

$$= \sum_{s=1}^{S} \sum_{n_s=1}^{N_s} \log\left(\lim_{T\to\infty} \frac{1}{T} \sum_{t\in T} \mathbb{E}\{u_{n_s}(t)\} \right). \tag{17}$$

IV. Proposed Solution

Our proposal consists on solving the previous stochastic problem by applying the *drift-plus-penalty* method described in [2]. This method allows us to build a new deterministic problem, which provides an approximate solution to the original one. Even more, the solution found can be made arbitrarily close to the optimal, but with a trade-off on the fulfillment of the constraints.

A. Problem Transformation

Because problem (12)-(15) consists on the optimization of a concave non-linear function of time averages, we need to transform it to apply the *drift-plus-penalty* method. For this

we follow the auxiliary variable technique of [2], to transform it into a classical time average optimization problem.

With this technique, the stochastic network optimization problem (12)-(15) can be transformed using a vector of auxiliary variables $\boldsymbol{\gamma(t)} = (\gamma_1(t), ..., \gamma_{N_s}(t))$ that are chosen every slot according to the constraints $0 \leq \gamma_{n_s}(t) \leq C_{n_s}^{max}$, where $C_{n_s}^{max}$ is the maximum possible transmission rate at each client and slice. The modified problem is:

$$\begin{aligned}
\underset{\boldsymbol{x}}{\text{maximize}} \quad & \overline{\phi(\boldsymbol{\gamma})} && (18)\\
\text{subject to} \quad & \overline{u}_{n_s} \geq 0 \; \forall s \in [1,S], \; \forall n_s \in [1,N_s], && (19)\\
& \overline{\gamma}_{n_s} \leq \overline{u}_{n_s} \; \forall s \in [1,S], \; \forall n_s \in [1,N_s], && (20)\\
& 0 \leq \gamma_{n_s}(t) \leq C_{n_s}^{max} \\
& \quad \forall s \in [1,S], \; \forall n_s \in [1,N_s], \; \forall t, && (21)\\
& \sum_{s=1}^{S} \sum_{n_s=1}^{N_s} x_{n_s}(t) \leq 1, && (22)\\
& 0 \leq x_{n_s}(t) \leq 1 \\
& \quad \forall s \in [1,S], \; \forall n_s \in [1,N_s], \; \forall t. && (23)
\end{aligned}$$

where

$$\begin{aligned}
\overline{\phi(\boldsymbol{\gamma})} &= \lim_{T\to\infty} \frac{1}{T} \sum_{t\in T} \mathbb{E}\{\phi(\boldsymbol{\gamma}(t))\} && (24)\\
&= \lim_{T\to\infty} \frac{1}{T} \sum_{t\in T} \mathbb{E}\left\{ \sum_{s=1}^{S} \sum_{n_s=1}^{N_s} \log(\gamma_{n_s}(t)) \right\} && (25)
\end{aligned}$$

Intuitively, we can explain the previous transformation as follows. Note that the original constraints are a subset of the new ones. Hence, any solution to the transformed problem will also satisfy the original constraints. Suppose we have decisions $\boldsymbol{x}^*(t)$, which are a solution to the original problem, augmented with the constraint

$$0 \leq u_{n_s}(t) \leq C_{n_s}^{max} \; \forall t. \tag{26}$$

Let $\overline{\boldsymbol{u}}^*$ be the expected utilities obtained by the clients under those decisions, which shield a maximum utility value $\phi(\overline{\boldsymbol{u}}^*) = \phi^{opt}$. Then, we can construct 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 (20), as we enforce equality, and constraint (21) because of the added constraint (26). 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. If the bounds in (26) are chosen large enough to contain a solution of the original problem, ϕ^{opt} equals the optimal value (note that this is our case, as each client utility is limited by the transmission rate). Therefore, a solution to the transformed problem ensures that the constraints of the original problem are satisfied, and it obtains an utility that approximates the original problem utility, as constraint (20) is forced to equality.

B. The drift-plus-penalty algorithm

To solve the problem (18)-(23) using the drift-plus-penalty method, we first transform the constraints into queue stability problems. For constraints (19) and (20) we define *virtual queues*, with update equation:

$$\begin{aligned}
Z_{n_s}(t+1) &= [Z_{n_s}(t) - u_{n_s}(t)]^+ \\
&\qquad \forall s \in [1,S], \; \forall n_s \in [1,N_s] && (27)\\
G_{n_s}(t+1) &= [G_{n_s}(t) + \gamma_{n_s}(t) - u_{n_s}(t)]^+ \\
&\qquad \forall s \in [1,S], \; \forall n_s \in [1,N_s] && (28)
\end{aligned}$$

where $[\cdot]^+ = \max\{\cdot, 0\}$.

These queues are virtual and do not represent real network queues. Intuitively, they can be seen as queues which accumulate the amount of bit rate and fairness not satisfied. By ensuring strong stability on these virtual queues, we guarantee that the constraints are met (for a complete demonstration see [2]).

Hence, the drift-plus-penalty strategy consists on minimizing the queue backlogs, as well as on minimizing an utility function called *penalty*. In our case the *penalty* is the opposite of the utility function $\phi(\boldsymbol{\gamma}(t))$.

For the queue backlogs, the following *Lyapunov function* is defined, as a measure of the length of both queues:

$$L(\boldsymbol{\Theta}(t)) = \frac{1}{2} \sum_{s=1}^{S} \sum_{n_s=1}^{N_s} Z_{n_s}(t) + \frac{1}{2} \sum_{s=1}^{S} \sum_{n_s=1}^{N_s} G_{n_s}(t) \tag{29}$$

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

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

Therefore, the objective is to minimize:

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

where V is a non-negative constant that will affect the trade-off between the drift and the penalty.

Finally, the strategy consists on minimizing the following upper bound of the previous expression:

$$\begin{aligned}
&\Delta(\boldsymbol{\Theta}(t)) - V\mathbb{E}\{\phi(\boldsymbol{\gamma}(t)) \mid \boldsymbol{\Theta}(t)\} \leq D \\
&\quad - V\mathbb{E}\{\phi(\boldsymbol{\gamma}(t)) \mid \boldsymbol{\Theta}(t)\} + \sum_{s=1}^{S} \sum_{n_s=1}^{N_s} G_{n_s}(t)\mathbb{E}\left\{\gamma_{n_s}(t)\right\} \\
&\quad + \sum_{s=1}^{S} \sum_{n_s=1}^{N_s} (Z_{n_s}(t) + G_{n_s}(t))\mathbb{E}\left\{-u_{n_s}(t) \mid \boldsymbol{\Theta}(t)\right\} && (32)
\end{aligned}$$

We can find an approximate solution to the problem by minimizing, on each slot t, the right-hand side of (32). For this, we separate the optimization in: (1) a minimization of the γ_{n_s} terms; and (2) a minimization of the x_{n_s} terms.

First, we find the optimal auxiliary variables, by considering $G_{n_s}(t)$ fixed:

$$\begin{aligned}
\underset{\boldsymbol{\gamma}(t)}{\text{minimize}} \quad & -V\phi(\gamma_{n_s}(t)) + \sum_{s=1}^{S} \sum_{n_s=1}^{N_s} G_{n_s}(t)\gamma_{n_s}(t) \\
\text{subject to} \quad & 0 \leq \gamma_{n_s}(t) \leq C_{n_s}^{max} \; \forall s \in [1,S], \; \forall n_s \in [1,N_s]. && (33)
\end{aligned}$$

We can find a closed-form solution for problem (33). First, we transform the single minimization problem into a multi-

ple problem by minimizing each sum term (remember that $\phi(\gamma_{n_s}(t)) = \sum_{s=1}^{S}\sum_{n_s=1}^{N_s}\log(\gamma_{n_s}(t))$):

For each slice $s \in [1, S]$ and for each client $n_s \in [1, N_s]$:

$$\begin{aligned} \underset{\boldsymbol{\gamma}(t)}{\text{minimize}} \quad & -V\log(\gamma_{n_s}(t)) + G_{n_s}(t)\gamma_{n_s}(t) \\ \text{subject to} \quad & 0 \leq \gamma_{n_s}(t) \leq C_{n_s}^{max}. \end{aligned} \tag{34}$$

Finding the derivative and making equal to zero we obtain:

$$\gamma_{n_s}(t) = \frac{V}{G_{n_s}(t)}. \tag{35}$$

Then, also for each slot t, we observe the values of the virtual queues $Z_{n_s}(t)$ and $G_{n_s}(t)$, and the current channel state $C_{n_s}(t)$, to find the $\boldsymbol{x}(t)$ that solves:

$$\begin{aligned} \underset{\boldsymbol{x}(t)}{\text{minimize}} \quad & \sum_{s=1}^{S}\sum_{n_s=1}^{N_s}(Z_{n_s}(t) + G_{n_s}(t))(-u_{n_s}(t)) \\ \text{subject to} \quad & \sum_{s=1}^{S}\sum_{n_s=1}^{N_s} x_{n_s}(t) \leq 1, \\ & 0 \leq x_{n_s}(t) \leq 1 \; \forall s \in [1, S], \; \forall n_s \in [1, N_s]. \end{aligned} \tag{36}$$

Using the fact that $u_{n_s}(t) = R_{n_s}(t) - K_s$, removing constants and changing the sign, the previous formulation can be transformed into (remember that $R_{n_s}(t) = C_{n_s}(t)x_{n_s}(t)$):

$$\begin{aligned} \underset{\boldsymbol{x}(t)}{\text{maximize}} \quad & \sum_{s=1}^{S}\sum_{n_s=1}^{N_s} C_{n_s}(t)(Z_{n_s}(t) + G_{n_s}(t))x_{n_s}(t) \\ \text{subject to} \quad & \sum_{s=1}^{S}\sum_{n_s=1}^{N_s} x_{n_s}(t) \leq 1, \\ & 0 \leq x_{n_s}(t) \leq 1 \; \forall s \in [1, S], \; \forall n_s \in [1, N_s]. \end{aligned} \tag{37}$$

Finally, we get a deterministic optimization problem that, at every slot t, observes the virtual queues $Z_{n_s}(t)$ and $G_{n_s}(t)$, the random channel capacities of each client $C_{n_s}(t)$, and finds the control actions $x_{n_s}(t)$ to satisfy (37). Note that the channel capacities and the virtual queue backlogs on time slot t act as constants in the maximization problem. After each iteration, the virtual queues are updated, as defined in (27) and (28). It can be proven that this solution satisfies all constraints in (18)-(23), and that the obtained utility differs from the target utility by no more than D/V, which can be made arbitrarily small as V is increased. However, the time average queue backlog bound increases linearly with V. This performance-backlog trade-off of $[O(1/V), O(V)]$ has been analyzed in [2]. In our case it translates into a trade-off between the optimal bit rate achieved and the satisfaction of the bit rate guarantees.

C. Scheduling Algorithm

A simplification of the previous problem is to consider that at each slot the AP is allowed to transmit to one client only. This would make the scheduling easier, as it does not need to calculate the ratios for each client at each slot, but only to decide to which client to transmit. Hence, the new optimization problem is:

$$\begin{aligned} \underset{\boldsymbol{x}(t)}{\text{maximize}} \quad & \sum_{s=1}^{S}\sum_{n_s=1}^{N_s} C_{n_s}(t)(Z_{n_s}(t) + G_{n_s}(t))x_{n_s}(t) \\ \text{subject to} \quad & \sum_{s=1}^{S}\sum_{n_s=1}^{N_s} x_{n_s}(t) \leq 1, \\ & x_{n_s}(t) \in 0, 1. \end{aligned} \tag{38}$$

It is easy to observe that the previous problem has the form of the *Knapsack Problem*, where each object or item n_s in time slot t gives a reward of $C_{n_s}(t)(Z_{n_s}(t) + G_{n_s}(t))$, where all items weight 1 and where the maximum capacity is also 1. It is simple to note that the solution of this problem is the item with the highest reward. Then, from the previous analysis we design the scheduling algorithm shown in Algorithm 1.

Algorithm 1 Scheduler with fairness

Input: V, K_s,$C_{n_s}(t)$, $\gamma_{n_s,max} \;\; \forall s \in [1, S], \; \forall n_s \in [1, N_s]$
Output: Client to schedule on each slot t

1: Initialize $Z_{n_s}(0) = 0$, $G_{n_s}(0) = 0$
2: **for all** time slot t **do**
3: For each slice and client calculate the auxiliary variables values $\gamma(t)_{n_s} = \min\{\frac{V}{G_{n_s}(t)}, \gamma_{n_s,max}\}$
4: Observe the current capacity of every client $C_{n_s}(t)$
5: Calculate the vector of benefits $\boldsymbol{U} = [C_{1_1}(t)(Z_{1_1}(t) + G_{1_1}(t)), ..., C_{n_s}(t)(Z_{n_s}(t) + G_{n_s}(t))]$
6: Find the client with the maximum benefit $i = arg\,max\,\boldsymbol{U}$
7: Schedule the client i to transmit in slot t ($x_i(t) = 1$).
8: Calculate $Z_{n_s}(t+1) = \max[D_{n_s}(t) - R_{n_s}(t) + K_s, 0]$ and $G_{n_s}(t+1) = \max[G_{n_s}(t) - R_{n_s}(t) + \gamma_{n_s}(t) + K_s, 0]$ $\forall s \in [1, S], \; \forall n_s \in [1, N_s]$.
9: **end for**

D. Time Slots in WiFi

In WiFi, nodes transmit randomly depending on the state of the channel (idle or busy), hence, the concept of time slots does not explicitly exist. Therefore, to be able to implement our proposal of scheduling on a WiFi node we need to simulate the time slots. Our proposal is to use the idea of *time quantums* and the *Adaptive Time-Excess Round Robin* (ATERR) algorithm, which we proposed in our previous work [13]. The idea is that each time the scheduling algorithm assigns a transmission opportunity to a client, as much data as possible is transmitted to that client for the duration of a *time quantum*. It is important to note that in WiFi, the transmissions are made in frames, and queues also store frames of data. Hence, it is very likely that the size of a *quantum* does not exactly match a given number of frames. The use of the ATERR algorithm thus becomes relevant, since the additional time consumed in one assignment is decremented from the next. By using ATERR with the same fixed quantum size for every client it becomes possible to have a slotted transmission, with the difference that slots are of variable size, but having a mechanism that yields, in the long run, the correct time assignment.

Even more, to implement the proposed scheduler, a queue per client and per slice is needed. This has already been proposed in our previous work [13] where we designed, analyzed, and implemented the necessary queuing structure.

E. Channel Capacity Estimation

Until now, we have assumed that we can know the exact value for the channel capacity $C(t)$ at each time slot. However, this is a complex task, which would need continuous monitoring at the AP and clients. Hence, our proposal is to use an estimation of this value, which can be obtained from the *rate control* mechanism of the AP. Nowadays, the most widely used mechanism is called *Minstrel* [15], which uses the frame loss rate to estimate the channel capacity. We will use this estimation, although not being optimal, as it has the advantage to be obtained from real data from the environment, with almost non overhead.

V. Guaranteeing Isolation

In the context of guaranteed bit rate, the isolation between slices and between clients within a slice is an important issue so as to keep the agreed guarantees regardless of the clients' behaviour. We consider two different types of isolation violations in the context of guaranteed bit rate: **Excess of offered traffic**, and **Lack of resources**.

The first case appears when the traffic of a client within a guaranteed bit rate slice exceeds the agreed maximum bit rate, and it thus consumes resources from other clients or slices. To prevent this, the slicing architecture must control and limit the traffic to conform the agreed characteristics in the corresponding SLA. However, it is also desirable to use resources efficiently, and to allow extra traffic if it does not affect other clients and slices. Hence, the control mechanism must also consider not to limit traffic if free resources are available. The second case emerges when there is not enough resources to provide the agreed guarantees to each client, thus affecting the 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 deteriorate after the initial connection, making the device to assign more resources to provide the agreed bit rate.

We propose to integrate the isolation management to the scheduler proposed in Algorithm 1. For the first case no changes are needed to the scheduler, as it implicitly limits the traffic through the scheduling, assigning to each client the requested resources. Even more, if free resources are available and there is excess traffic, it could use those resources. This happens because, once the virtual queues are stable (all guarantees are satisfied), the scheduler continues to schedule the clients to maximize the utility, allowing them to transmit the exceeded traffic. On the other hand, to attack the isolation issue when the channel conditions of a client deteriorates and more resources than available are necessary, an extension to the scheduler is needed. We propose a solution in two steps: monitoring and action.

We propose to monitor the evolution of the virtual queues. Remember that the virtual queues $Z_i(t)$ model the bit rate constraints, and they can be conceived as buffers of the amount of bit rate not 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 being satisfied. Then, when we detect a situation of a constant increment on the size of any of the virtual queues, we can take the appropriate action.

For the second step, our solution to solve the detected isolation issue is to degrade the performance or even to disconnect some clients on a controlled way. 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 this work, we introduce a simple strategy for the actions to take. We propose to select a client and to remove its bit rate guarantee, downgrading it to a *best-effort* client. This solution must be accompanied with a message to the service provider to take appropriate action. With this solution we allow the scheduler to assign resources to the client, but only if they are available. To decide which client to downgrade we propose to select the slice with the lowest bit rate requirement, and within this slice, to choose the client with the highest use of resources. If downgrading the selected client does not solve the isolation issue, we continue downgrading clients until it is resolved. The use of this policy is arbitrary, and other options are perfectly suitable.

VI. Numerical Results

We used MATLAB to assess the performance of the proposed solution, and we evaluated different scenarios to validate its correct behaviour. The setup for all the evaluation scenarios consists on one WiFi AP, where three slices are instantiated:

- Slice 1: Guaranteed bit rate of 5 Mbps to each client.
- Slice 2: Guaranteed bit rate of 3 Mbps to each client.
- Slice 3: Guaranteed bit rate of 2 Mbps to each client.

We tested the proposed scheduler with fairness in scenarios with different resource usage, both with stable and variable channel capacities

A. Scenario 1: Complete resource usage

In this first experiment the objective is to test a scenario where all available resources are needed to satisfy the slice requirements. We consider three clients with fixed positions, with the following stable channel capacities between the clients and the AP: Client 1: 20 Mbps; Client 2: 6 Mbps; Client 3: 8 Mbps.

Each slice has only one associated client: Client 1 belongs to Slice 1, Client 2 belongs to Slice 2 and Client 3 belong to Slice 3. In the simulation, the traffic is generated at a constant bit rate, yielding full buffer/saturation conditions. It is easy to observe that due to slice requirements, all the available capacity must to be assigned to guarantee the requests. In particular, Client 1 will use $1/4$ of the airtime, Client 2 $1/2$, and Client 3 $1/4$. In Figures 1 and 2 we show the instantaneous throughput observed at each client during the whole experiment and the size of the virtual queue, respectively. The results highlight that, after a small convergence period, virtual

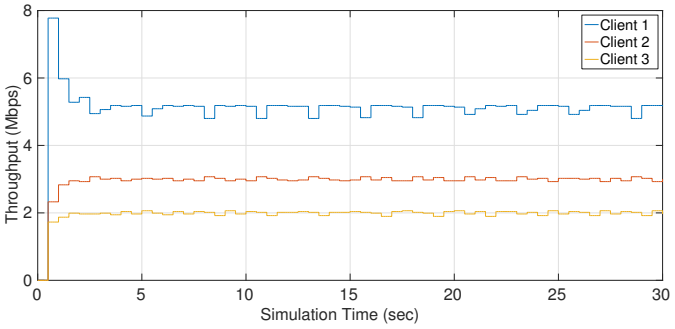

Fig. 1. Throughput in Scenario 1

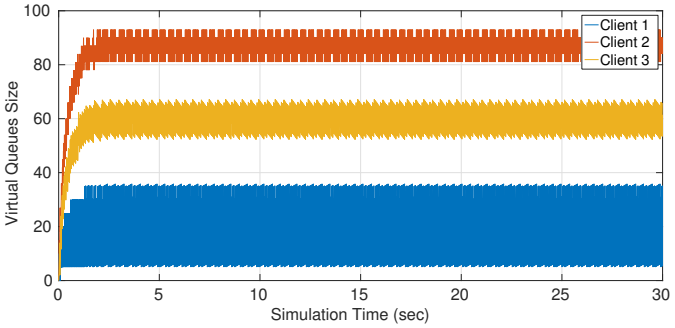

Fig. 2. Size of Virtual Queues $Z(t)$ in Scenario 1. There is a virtual queue per client, which accumulates the difference between the required minimum bit rate and the achieved bit rate by such client.

queues are stabilized, and the solution provides the guaranteed bit rate to each client, as required by the slices. The algorithm is configured with the parameter $V = 1$ which guarantees a fast convergence. In the following scenario we experiment with different values of V.

B. Scenario 2: Extra resources available

This scenario is similar to the previous one, but the channel capacity of Client 1 is increased to 30Mbps at second 10 of the simulation. Then, at second 20 the capacity returns to 20Mbps. This variation generates a surplus of resources that, if slices have additional traffic, can be efficiently used so as to improve performance. With the increased capacity, the transmission to Client 1 consumes $1/6$ of the total airtime. Client 2 consumes $1/2$ of the total airtime and Client 3 consumes $1/4$. Hence, there is $1/12$ of the airtime unused. In the case of a perfect proportional fair sharing, each client is allocated a third of this fraction. Therefore, the obtained throughput of Clients 1, 2, and 3 would be 5.83 Mbps, 3.16 Mbps and 2.22 Mbps respectively.

As we mentioned previously, by solving the problem with the drift-plus-penalty method, there is a trade-off between reaching the optimum and satisfying the constraints. This trade-off is governed by the parameter V introduced in (31). In this case, as V increases we are closer to a fair allocation, but it becomes more difficult achieving the bit rate guarantees.

We show the results obtained with the fairness scheduler with three different values of V. In Figures 3 and 4 we show the throughput and virtual queue sizes if $V = 50$. When the capacity of Client 1 gets higher, all three clients increase their throughput. At $t = 10$, Client 1 gets a higher channel capacity

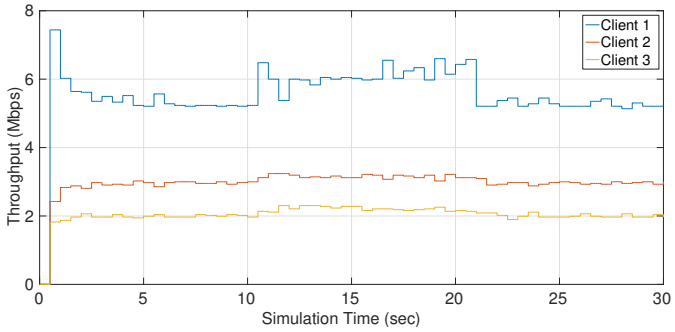

Fig. 3. Throughput in Scenario 2 with $V = 50$.

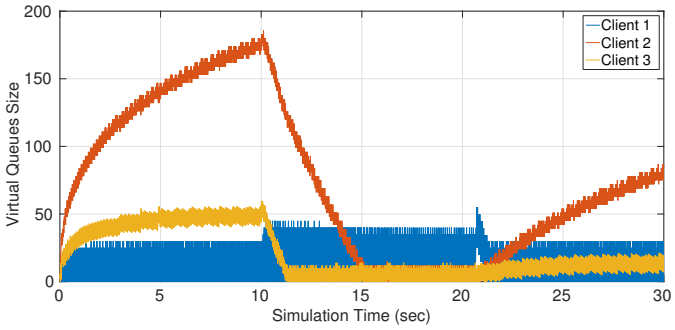

Fig. 4. Size of Virtual Queues $Z(t)$ in Scenario 2 with $V = 50$.

and it can thus transmit its data faster, yielding more resources. One can observe that the throughput values obtained are very close to the optimum ones. However, from Figure 4 we can observe that virtual queues for Clients 2 and 3 struggle to converge when the resources are tight. We also evaluate the algorithm with a value of $V = 1$ and $V = 500$ in Figures 5 and 6. As expected, with $V = 1$ the guarantees are reached very fast, but there is no fairness in the assignment. On the other hand, with $V = 500$ Client 2 and 3 never achieve the bit rate guarantees, but the allocation is almost optimal in fairness.

C. Scenario 3: Lack of resources

In this scenario the channel capacity of Client 1 is set to 10 Mbps from second 15 until the end of the simulation.

Note that this scenario generates a case of isolation violation as any possible resource allocation would accomplish the required guarantees (there is no feasible solution to the optimization problem). Hence, we can observe in Figures 7 and 8 that, when the channel of Client 1 deteriorates, the throughput

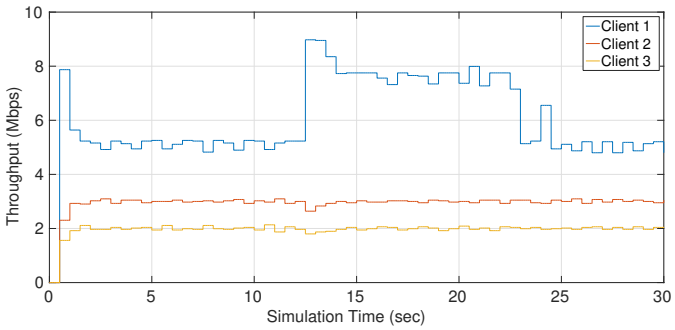

Fig. 5. Throughput in Scenario 2 with $V = 1$.

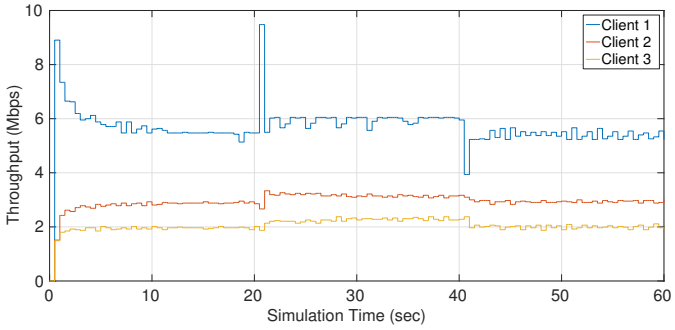

Fig. 6. Throughput in Scenario 2 with $V = 500$.

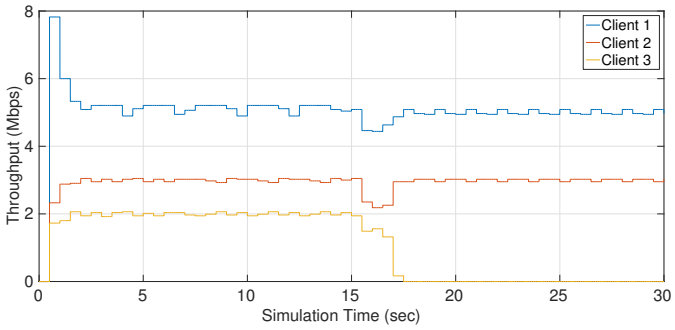

Fig. 7. Throughput in Scenario 3.

of the three clients is affected and the virtual queues start growing. However, since we implement the aforementioned detection and isolation scheme, our solution is able to take the necessary actions to continue providing the guarantees to some of the slices. In this case (see Figure 7), Clients 1 and 2 still get their guaranteed bit rate, while Client 3 is disconnected. Note that, as we mentioned in Section V, the scheduler follows the policy to disconnect the client with the highest use of resources from the slice with the lowest bit rate requirement.

VII. Conclusions

We have proposed a resource allocation mechanism for guaranteed bit rate slicing in WiFi Access Points. The mechanism controls the scheduling of the transmission time to the different slices to satisfy minimum bit rate demands. We have also included a simple approach to detect and control guarantees violations. Moreover, we implemented a prototype of our solution, and we validated its main functionality of providing bit rate guarantees.

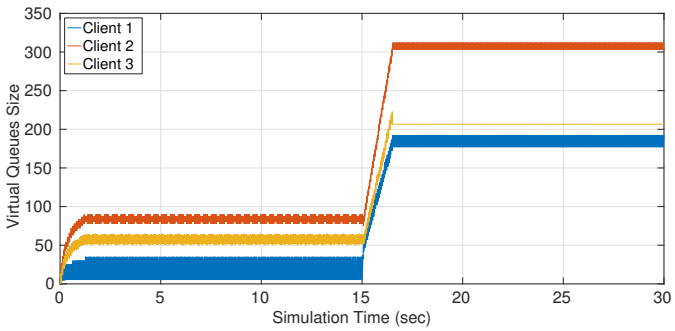

Fig. 8. Size of Virtual Queues $Z(t)$ in Scenario 3.

In the context of slicing, several existing works have introduced frameworks or high-level management solutions, but there is an absence of ideas on how to specifically implement slicing at the WiFi APs. Our objective with this proposal is to overcome this issue and allow WiFi to be integrated to the slicing paradigm.

The initial results have shown to be promising, but additional experimentation is still needed. We are currently working on implementing our solution on a network simulator to evaluate more complex scenarios. In future research we also plan to extend the mechanism to provide delay guarantees.

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

[2] 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.

[3] 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.

[4] A. Banchs and X. Perez, "Providing throughput guarantees in ieee 802.11 wireless lan," in *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, vol. 1. IEEE, 2002, pp. 130–138.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] M. Richart, J. Baliosian, J. Serrat, 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, Ed. IEEE, 2017, In press.

[11] 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.

[12] 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.

[13] 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*, pp. 1–31, 2018.

[14] F. P. Kelly, A. K. Maulloo, and D. K. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research society*, vol. 49, no. 3, pp. 237–252, 1998.

[15] Minstrel rate control algorithm. [Online]. Available: https://wireless.wiki.kernel.org/en/developers/documentation/mac80211/ratecontrol/minstrel