Journal of Network and Systems Management manuscript No.
(will be inserted by the editor)

Slicing in WiFi Networks Through Airtime-based Resource Allocation

Matías Richart · Javier Baliosian · Joan Serrat · Juan-Luis Gorricho · Ramón Agüero

Received: date / Accepted: date

Abstract Network slicing is one of the key enabling technologies for 5G networks. It allows infrastructure owners to assign resources to service providers (tenants), which will afterwards use them to satisfy their end-user demands. This paradigm, which changes the way networks have been traditionally managed, was initially proposed in the wired realm (core networks). More recently, the scientific community has paid attention to the integration of network slicing in wireless cellular technologies (LTE). However, there are not many works addressing the challenges that appear when trying to exploit slicing techniques over WiFi networks, in spite of their growing relevance. In this paper we propose a novel method of proportionally distributing resources in WiFi networks, by means of the airtime. We develop an analytical model, which shed light on how such resources could be split. The validity of the proposed model is assessed by means of simulation-based evaluation over the `ns-3` framework.

Keywords Wireless network slicing · Wireless resource management · Airtime allocation · Queue scheduling · Deficit round robin · 5G · WiFi

Matías Richart
School of Engineering, University of the Republic, Uruguay.
Tel.: +598-27142714
E-mail: mrichart@fing.edu.uy

Javier Baliosian
School of Engineering, University of the Republic, Uruguay.
E-mail: javierba@fing.edu.uy

Joan Serrat
Network Engineering Department, Polytechnic University of Catalonia, Spain.
E-mail: serrat@tsc.upc.edu

Juan-Luis Gorricho
Network Engineering Department, Polytechnic University of Catalonia, Spain.
E-mail: juanluis@entel.upc.edu

Ramón Agüero
University of Cantabria, Spain.
E-mail: ramon@tlmat.unican.es

1 Introduction

In the last years there has been a trend of increasing heterogeneity in wireless access networks. The heterogeneity is not only on the deployment of cells of different sizes but also on new types of communications, diverse access technologies and diverse offered services. It is expected that in the new generation of wireless networks (5G) the heterogeneity increases, requiring new management and control techniques to cope with this diversity. For example, 5G wireless networks will need to provide service to a variety of different applications and use-cases employing for this purpose diverse technologies and equipment. Hence, the network infrastructure will need to deal with a diversity of traffic patterns, service requirements and devices capabilities. It is envisioned that, in this scenario, an efficient use of network resources can be achieved by individually managing and controlling each use-case [1].

Therefore, the *network slicing* concept has been proposed as the paradigm which partitions a shared physical network infrastructure into multiple self-contained logical pieces (slices) established to meet certain network characteristics. In short, this paradigm permits to create dynamic and on-demand resource slices to cope with specific service requirements. This is the key difference from existing similar proposals such as VPNs, but a characteristic that makes it complex to deploy and manage.

Recent literature has proposed *network slicing* as a main enabler for future 5G networks. As explained in [2], network slices leverage deploying services with contradictory requirements over a shared infrastructure, easing the management of the network. Particularly, slicing the network allows to individually configure the networks *edge-to-edge* and define specific functions for each case, while sharing the same infrastructure. Moreover, in [3] slicing is proposed as one of the key capabilities of 5G to manage the expected heterogeneous requirements of future mobile networks.

In this context, infrastructure providers can lend their network resources to new business players, such as virtual mobile network operators (VMNOs) or Over-The-Top (OTT) applications. These novel players act as *tenants* of the infrastructure and are given complete control over the lent resources. Therefore, delivering efficient resource allocation as well as guaranteeing isolation are the main challenges of this paradigm. Hence, the design of effective resource allocation policies and mechanisms is an important research issue in network slicing. Most of current research has focused on the network core and on the definition of frameworks to enable slicing. More recent works focus on applying slicing only to cellular networks and, in particular, for the LTE technology.

In this work we make a proposal to solve the resource allocation problem that enables slices on the IEEE 802.11 (WiFi) technology, as an evolution from our previous work [4]. In this paper, we incorporate an analytical study of the proposed solution as well as a more exhaustive evaluation. In particular, we concentrate on a variant named *Infrastructure-Sharing Slicing*. Its objective is to split and allocate network resources into slices, proportionally to the requested demand (see Section 3).

More precisely, we propose a novel mechanism for slicing a WiFi Access Point (AP) by considering the transmission time (airtime) as the resource to share. Our resource allocation proposal is efficient, as each slice only receives the exact amount of resources required for its current load, so free resources can be used by other slices.

Furthermore, we theoretically analyze the proposed mechanism so as to find the necessary conditions for an accurate allocation of airtime. In this analysis we also show that our proposal fairly allocates airtime among users of the same slice. We also identify the parameters that influence the latency caused by the scheduling algorithm. Finally, we complete the analysis with simulated experiments, where we validate the theoretical results and show the behaviour of our mechanism in an illustrative use case.

The paper is organized as follows: in Section 2 we review the state of the art in wireless slicing, in Section 3 we introduce our model of network slicing and define the concept of *Infrastructure-Sharing Slicing*; in Section 4 we describe the airtime slicing proposal in detail. Section 5 depicts the analytical evaluation, while Section 6 discusses the evaluation of our proposal. Finally, Section 7 concludes the paper, and identifies our future work.

2 Related Work

The idea of controlling the airtime usage in WiFi has been mainly studied as a means to overcome the *performance anomaly* problem [5]. This problem appears in multi-rate wireless local area networks (WLANs) and causes the transmission throughput of any station to be bounded by the lowest rate of all stations associated to the AP. Hence, several works have proposed different mechanisms for a fair allocation of the airtime among the stations associated with the same AP (see [6, and references therein]). In particular, works [7,8,6] have suggested the use of a modified Deficit Round Robin (DRR) [9] scheduling to achieve fairness. As will be explained in Section 4, our proposal adapts the airtime scheduling mechanism from [6] to be used for network slicing. The work in [6] showed promising results on achieving airtime fairness for solving the *performance anomaly* problem, including an implementation on real hardware (*Atheros ath9k* driver).

Network slicing is a rather new concept but many of the challenges for resource allocation have already been considered extensively in the domain of network virtualization. Existing proposals for resource allocation in wireless network virtualization and slicing are highly determined by the wireless technology. Therefore, most of the literature concentrates on the 3GPP LTE or IEEE 802.11 standards. As our approach is focused on the IEEE 802.11 technology we concentrate on the related work in this technology.

Most of current works which deal with resource allocation for slicing (or virtualization) consider traffic flows as the entities to assign resources. In those works the goal is to share physical devices at the flow-level. In this context, [10] introduces the idea of Virtual Access Points (VAPs), which will be used by many following works on this area. For example, [11,12] extended and improve the idea of virtualizing a physical interface into multiple virtual APs. The objective of this virtualization is to allow sharing one physical device among several virtual networks. The main idea is that each VAP uses different SSIDs and allows to manage association and transmission parameters for each virtual network. However, all VAPs share the same physical interface which implies that they share the physical transmission parameters such as the frequency (channel).

Following the classification introduced in our previous work [13], we analyze existent works on resource allocation for slicing in three categories:

- *EDCA control*: Solutions in this category adjust the EDCA parameters (*Contention Window*, *Arbitration inter-frame spacing* and *Transmission Opportunity*) of the MAC layer so as to control how the slices access to the wireless medium.
- *Slice scheduling*: These strategies consist on scheduling resources usage among the slices. For example, scheduling the transmission opportunities.
- *Traffic shaping*: These are the solutions which shape the slices traffic so as to conform performance requirements.

In the *EDCA Control* category we can mention four works which propose a resource allocation solution for slicing. In [14] is proposed a mechanism to guarantee throughput fairness among VAPs. The goal of this work is to distribute throughput fairly among VAPs regardless of the number of associated clients. It uses control-theory (a Proportional Integral (PI) controller) to control the *Contention Window* (*CW*) parameter of each client of the VAPs so as to achieve the fairness objective. However, it does not provide any type of guarantees of throughput or airtime. The authors in [15] proposed an airtime resource allocation method through the management of the *Contention Window*. In this work each slice is represented by a VAP with its own MAC queues and is able to control its individual set of EDCA parameters. Each slice defines its target airtime ratio and an allocation algorithm adjusts the *CW* to achieve those ratios. The proposal requires several modifications on the devices to implement the virtual APs. A similar idea to that found in [15] is presented in [16] for the uplink traffic, and a slice scheduling mechanism for the downlink traffic. The goal is also to provide airtime share among *groups* of users and to maximize system throughput. The uplink control mechanism configures two EDCA parameters, the *Contention Window* and the *Transmission Opportunity* for each client following an analytical result. For the downlink, a scheduling-based airtime allocation scheme is addressed. It fosters a similar approach to our proposal, although airtime scheduling is not its main focus and the algorithm description is very vague. The scheduler works in two stages, first it schedules packets per-slice following the shares of each slice and, secondly, it schedules packets per-user using a Round Robin scheduler. The work in [17] presents a mechanism that allows sharing a wireless network (WLAN) with multiple APs among several virtual networks. Its objective is to maximize the throughput of the entire network while guaranteeing a minimum airtime to each virtual WLAN. It also focus on uplink traffic controlling only the behaviour of the clients. Differently from other works, in this case the problem is formulated as an optimization problem where association and airtime are jointly controlled so as to maximize throughput while airtime constraints for each virtual network are respected. The solution proposed involves adjusting 5 different MAC parameters such as the *Contention Window* and the *Arbitration inter-frame spacing* based on an approximate algorithm which solves the optimization problem. The work lacks of a description on how this could be implemented in wireless devices. For example, it seems that the optimization solution is executed centrally, hence it would be necessary to have a mechanism to inform the parameters to the clients.

The work of Katsalis et. al. [18] falls in the *Slice Scheduling* category. In this work it is proposed a scheduling mechanism with feedback control so as to guarantee throughput ratios among slices. The idea is to split the total transmitted bytes of an AP into ratios requested by the different slices. This work proposed a queue structure implemented in a software router over the MAC layer of an AP and uses a packet scheduling mechanism to prioritize traffic. The scheduling consists on speeding up the traffic of slices which are below the guaranteed throughput and slowing down the flows which are above the guarantee. It uses feedback from the driver so as to account for the current transmitted bytes.

Works in [19] and [20] suggests to use *traffic shaping* to provide airtime guarantees to slices. In [19] it is proposed to use traffic shaping in Wifi so as to limit the use of resources from each slice by controlling the uplink airtime usage. The solution concentrates only on the client side and is based on commands sent by the AP so as to control the traffic. The uplink airtime usage of each slice is estimated at the AP using information sent by the clients by a preinstalled control and reporting module. Then, the airtime is controlled by a traffic shaping module also installed at each client following commands issued by the AP. The same authors present in [20] a traffic shaping technique for controlling downlink traffic in the WiMAX technology. The mechanism (called Virtual Network Traffic Shaper (VNTS)) shapes traffic using feedback information about the current transmission rate at the wireless interface, the number of clients and the weight of the slice. The shaper is located outside the Base Station so as to adapt the load offered to the frame scheduler with the objective of guaranteeing the assigned ratios of each slice.

The adaptation of EDCA parameters is a well investigated approach in WiFi virtualization and provides good results. However, it can be implemented either by modifying the MAC layer or only on predefined traffic classes (Access Categories defined in the IEEE 802.11 standard) which limits its application for slicing, where an arbitrary number of slices can be defined. Higher level approaches like traffic shaping or slice scheduling avoid these issues but present other drawbacks such as not considering queue buildup at the MAC or physical layers which can jeopardize the scheduling made at upper layers. Also, traffic shaping may require a complex feedback control to gather information from lower layers so as to use resources efficiently. A more extensive review on existing proposals for slicing can be found in our previous paper [13]. Furthermore, in [21] is presented a comprehensive review of wireless virtualization techniques.

As can be observed, airtime allocation has already been proposed for network slicing, however many unsolved challenges still remain. With our solution we aim to overcome those challenges and provide a solution for airtime allocation that can be implemented in WiFi devices. The main contributions and differences from previous work are:

- Our solution does not need to control low-level MAC parameters, neither it needs feedback from the wireless channel to achieve the required allocation. It only needs information about the consumed airtime, which can be obtained from the hardware driver.
- We do not require traffic shaping, which, if not controlled properly, can lead to a waste of unused resources.
- Our queuing model takes into account the hardware behaviour, to avoid queue buildups at lower layers, and allows packet aggregation.

- We extensively describe our scheduling algorithm and queuing model, allowing it to be implemented and the evaluations replicated.
- We theoretically demonstrate the characteristics and limits of our solution, finding the necessary conditions to guarantee an airtime allocation with a given tolerance.
- We implement our solution in a simulator, which allows to resemble real devices and we also make our simulation code available, so that the community would be able to freely use it.

3 Network Slicing

Although there are different definitions of *network slicing*, in this paper we use the definition introduced in our previous work [13], where a slice is a set of traffic flows with some common features. In this scenario 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. A slice is allocated by a *slice provider* and owned by an external entity (a *tenant*), and can include performance requirements. This may require a fraction of network resources to be reserved for a slice. A slice can support flows from multiple *final users* (mobile clients of the network in our case), but at the same time, a final user can participate in multiple slices. However, a flow belongs to a single slice, and slices are always independent between each other.

Some slice examples are: all the flows produced by the same type of device as source or destination (e.g., sensors); the flows of a VoIP service; the flows from or to a user of a given operator, etc.

A key aspect of slicing is *isolation*. Isolation ensures slices not to be disturbed by each other, avoiding performance degradation or misusing resources allocated to other slices.

We classify slicing in two variants:

- *Quality-of-Service Slicing (QoSS):* slices supporting different services and ensuring their Quality of Service, regardless of the required resources.
- *Infrastructure-Sharing Slicing (ISS):* similar to the idea of network virtualization, a set of resources are allocated for the exclusive use of a tenant. The tenant (e.g. Mobile Virtual Network Operator) has complete control over the network infrastructure and network functions within the slice.

QoSS variant requires the slice provider to seek performance objectives for each flow in the slice, while in *ISS* the tenant requires a set of resources to be allocated for the whole slice. Hence, the implementation details of these two types of slicing can vary significantly.

The previous definition applies for slicing in general. However, as already mentioned, in this work we focus on the slicing problem at the wireless edge and, particularly, on the *Infrastructure-Sharing Slicing* variant.

3.1 Infrastructure-Sharing Slicing

The main motivation for implementing ISS is an efficient use of wireless resources. Sharing the wireless resources among users, service providers, or network operators, contributes to the utilization of idle resources when possible, reducing the need of broadening network deployments.

In WiFi, it is becoming more popular to share residential APs for public use. Private companies and service providers offer the possibility to share their WiFi APs, opening them as hot-spots for public use. In this scenario, ISS can provide the necessary tools to achieve efficiency and guaranteed isolation. Moreover, ISS can promote inter-provider sharing, where each slice can be used by a different operator, offering its own services.

When deploying the ISS variant in the wireless domain, various challenges need to be tackled. As already mentioned, ISS has many similarities with network virtualization in the wired domain where the slice requests do not specify performance objectives for the slice flows, but demand resources to be allocated to an entire slice. For the wired domain it is a common practice to allocate resources such as link bandwidth or CPU usage. This is possible because the available resources (such as link capacity) are constant and known beforehand. However, the bandwidth of a wireless link is uncertain, as it depends on variable channel characteristics. As a consequence, the available resources to be assigned are limited, and constrained to: the available radio spectrum (divided in time, frequency or space) or the available transmission time (airtime).

Different techniques for allocating radio spectrum are used in several LTE slicing proposals, extending the usual scheduling of LTE Physical Resource Blocks (PRBs) [22]. For example, a tenant can request a given quantity of PRBs, or it can demand a fraction of the total PRBs available from a Base Station. Then, a PRB scheduler will allocate the feasible PRBs for every slice transmission. However, applying this approach to WiFi is not currently possible because of the differences on the use of the radio spectrum. In WiFi there is no time or frequency division, and every node transmits using a CSMA/CA approach, utilizing the entire assigned frequency band.

Hence, for WiFi, transmission time (or airtime) sharing is the only possible alternative. Airtime sharing implies distributing the transmission time, assigning a fraction to each slice. Nevertheless, this approach also presents a number of difficulties. For example, the airtime consumed by a given communication depends on its transmission rate and the retransmissions made by the MAC layer. Therefore, a device with a given amount of airtime to use can achieve different throughput performances, depending on the current channel characteristics. The scheduling of transmissions among the final users within a slice is also a challenge. It is expected that, while guaranteeing the airtime of the entire slice, each of its users receives a fair amount of this airtime. This is hard to achieve, considering that each user can operate at different data rates.

4 Airtime Slicing Proposal

As already mentioned, our focus is on the network's edge, more precisely, on WiFi Access Points. The goal is to implement *Infrastructure-Sharing Slicing* by allocating the requested airtime to the different slices. The model is simple, the slice provider commits to reserve a percentage of the total airtime in an AP to a particular slice, given that enough resources are available.

As said before, we characterize a slice by a set of flows, therefore, when created, a slice is setup specifying the flows that will belong to it. They may be enumerated (e.g. a list of source IP's and ports) or described by some common feature (e.g. VoIP traffic).

4.1 Adaptive Time-Excess Round Robin

We propose a queuing structure and a scheduling mechanism for airtime slicing, inspired on the work presented in [9] and [23] and its extension proposed in [6] for airtime fairness among clients of a WiFi AP. We call our mechanism *Adaptive Time-Excess Round Robin* (ATERR).

ATERR is based on (1) identifying the existing traffic flows, (2) assigning a queue to each flow, and (3) scheduling the service of each queue with a given criteria. The queues are serviced following a round-robin scheduling, which depends on a given *quantum* of time. The *quantum* is a configurable parameter that controls how much airtime is allocated to each queue in a round. When a packet is dequeued, the difference between the airtime consumed by the packet and the quantum is kept in a variable called *time-excess*. Packets are dequeued while a negative *time-excess* remains, and when it reaches a positive value, no more packets are dequeued and the algorithm moves to the next queue in a round-robin manner. On every new round, each queue's *time-excess* is updated with the previous *time-excess* value minus its *quantum*. This grants a direct control over the airtime used by each flow, regardless of the packet sizes or the transmission rate. However, since MAC layer may perform packet-aggregation and packet-retransmissions after a packet is dequeued, the actual airtime consumed is unknown beforehand. Then, the *time-excess* of a queue needs to be updated after each packet transmission. This may cause an allocated airtime "excess" in a round.

This allocation strategy does not guarantee instantaneous satisfaction of the requested share, but it does try to statistically allocate a certain airtime share in a given time window (which we call *allocation window*). A study of the size of this window is described in Section 5.

In summary, the main differences between ATERR and the mentioned previous works are:

- The goal of ATERR is to allocate the different airtime requests of each slice while in previous works the objective is to guarantee airtime fairness.
- Because of the specific characteristic of wireless transmissions, ATERR considers transmission excess instead of deficit.
- Quantums are adaptive and are computed based on the requested airtime share, the current number of slices, and the clients per slice. This way we support slice and clients connections and disconnections dynamically.

4.2 System Architecture

ATERR is envisioned to be deployed onto the APs of a WiFi network replacing the queuing and scheduling algorithms of the AP.

ATERR maintains a queue for each user's traffic flow within a slice (which coincides with a WiFi device). Hence, we have a queue per user and per slice. This

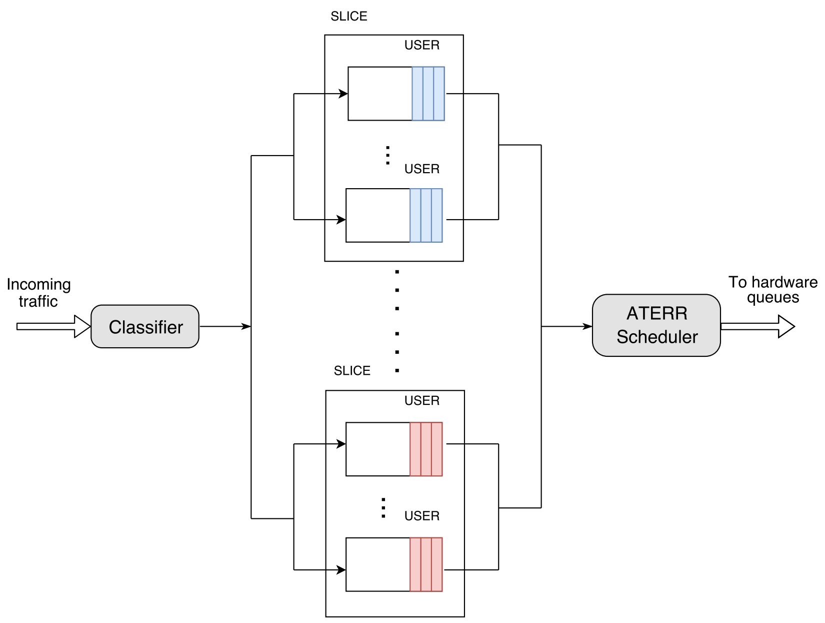

Fig. 1 ATERR simplified queuing architecture.

means that if a user participates in three slices, three queues will be created in the system for such user, one per slice (see Figure 1).

For our round robin algorithm, ATERR keeps a list of queues with backlogged traffic (*active queues*), which are the ones that need to be served. In particular, there are two lists of active queues: *newQueuesList* and *oldQueuesList*. Dealing with two different lists is a contribution from the FQ-CoDel algorithm [24] (also used in [6]), so as to reduce the latency of sparse flows, giving priority for transmissions of flows that do not build-up a queue. The reader might refer to [24] for a detailed description of its operation.

Furthermore, we add an *Inactive* state so as to identify queues without traffic. If a queue is in the inactive state for a predefined period of time (1 second in our implementation) it is removed from the slice and the quantums are recalculated. The queues are not removed immediately after being empty so as to soften quantum changes. The complete rationale is presented in Algorithm 1.

4.3 System Design

The design of ATERR follows two main objectives:

- To allocate the share of airtime requested by each slice.
- To allocate airtime to each user within a slice fairly.

We propose airtime fairness among users within a slice so as to avoid the *performance anomaly* problem mentioned in Section 2. This way, the slice resources are allocated fairly independently of the channel characteristics of the users.

Algorithm 1: ATERR scheduling pseudocode. For every new transmission the mechanism selects the queue (there is a queue per station and slice) to dequeue a packet from. It is a modified version of Algorithm 3 from [6].

```
function ScheduleTransmission() is
    input  : newQueues as the list of new active queues.
             oldQueues as the list of old active queues.
    output: queue as the queue with packets to transmit.
    if newQueues is not empty then
        queue ← GetFront(newQueues);
    else if oldQueues is not empty then
        queue ← GetFront(oldQueues);
    else
        return;
    end
    if queue.excess ≥ 0 then
        queue.excess ← queue.excess − queue.quantum;
        if queue ∈ newQueues then
            RemoveFront(newQueues);
        else
            RemoveFront(oldQueues);
        end
        InsertTail(queue, oldQueues);
        goto 2;
    end
    if queue is empty then
        if queue ∈ newQueues then
            RemoveFront(newQueues);
            InsertTail(queue, oldQueues);
        else
            RemoveFront(oldQueues);
            setQueueInactive(queue);
        end
        goto 2;
    end
    setAvailableToLowerLayer(queue);
end
```

To accomplish both objectives, the quantum of each queue must be dynamically computed based on the requested resources and the system state. In the following we explain how we compute the quantums. The notation used throughout the paper is defined in Table 1.

Lets define $\mathcal{S}$ as the set of all slices instantiated on an AP and $N_{\mathcal{S}}$ the number of slices in $\mathcal{S}$. Each slice $s \in \mathcal{S}$ requires a ratio of the AP's airtime to be allocated, denoted by $p_s \mid 0 < p_s \leq 1$. Lets also define $\mathcal{U}$ as the set of users associated with the AP, $\mathcal{U}_j$ the subset of those users that belong to slice s, and N_s the number of users in $\mathcal{U}_s$.

To achieve fairness among users of the same slice, all the quantums within the slice must be equal. Then, we denote by q_s the quantum of any queue in the slice s and define Q_s as the sum of all the quantums of slice s ($Q_s = \sum_{i=1}^{N_s} q_s = N_s q_s$).

To achieve the first objective, Q_s must satisfy:

$$Q_s = p_s \sum_{j=1}^{N_S} Q_j \quad \forall s \in \mathcal{S} \tag{1}$$

For the second objective, as all quantums of a slice need to be the same, the quantum of any queue in a slice s that requests a share of the airtime p_s and has N_s users is:

$$q_s = \frac{Q_s}{N_s} \quad \forall s \in \mathcal{S} \tag{2}$$

Substituting 1 in 2 we have:

$$q_s = \frac{p_s \sum_{j=1}^{N_S} Q_j}{N_s} = \frac{p_s \sum_{j=1}^{N_S} N_j q_j}{N_s} \quad \forall s \in \mathcal{S} \tag{3}$$

However, q_s appears on both sides of the equation, then isolating q_s we get:

$$q_s = \frac{p_s}{N_s}\left(\sum_{\substack{j=1\\ j\neq s}}^{N_S} N_j q_j + N_s q_s\right)$$

$$q_s = \frac{p_s}{N_s}\sum_{\substack{j=1\\ j\neq s}}^{N_S} N_j q_j + \frac{p_s}{N_s} N_s q_s$$

$$q_s = \frac{p_s}{N_s(1-p_s)}\sum_{\substack{j=1\\ j\neq s}}^{N_S} N_j q_j$$

Finally, we get that the quantum for each queue of the system must be:

$$q_s = \frac{p_s \sum_{j=1, j\neq s}^{N_S} N_j q_j}{N_s(1-p_s)} \quad \forall j \in \mathcal{S} \tag{4}$$

The above result shows the adaptive nature of the algorithm as all queue's quantums need to be recalculated every time a queue is created or removed (i.e. when a user connects or disconnects) on the AP. Even more, it is important to notice that the previous result does not provide an absolute value for the quantums, but a relationship between all the slice's quantums. To solve this indeterminate system of equations, we fix the minimum allowable quantum in the system and assign it to the lower q_s.

A remarkable feature of ATERR resides in that it does not impose a static assignment of resources, neither it limits the use of resources, but appropriately schedule packets (if available) granting the airtime assignment. For example, if a flow does not use all its assigned airtime, the unused airtime can be consumed by the other flows proportionally. So far, our current proposal only applies to downlink transmissions, but it is possible to extend it to consider also the uplink traffic.

Finally, it is important to point out that the proposed solution is implementable in current 802.11 hardware, provided that the developer has access to the MAC layer. For example, we argue that our solution can be implemented in the *Atheros ath9k* driver which is used in many current commercial off-the shelf Access Points. As mentioned in Section 2, the work in [6] has already implemented in this driver individual queues per client and an airtime scheduler. For our solution, we would need to add a mechanism for specifying the slices, redefine the queue structure to consider the slices, and implement the algorithm to dynamically calculate the quantum's values.

Table 1 Notation

Notation	Definition
$rounds$	Service opportunities received by a flow in a time interval. It can be seen as the round of the {round robin} algorithm.
m	Number of rounds in a given time interval (t_1, t_2).
$e_{i,s}(k)$	Value of the excess of queue i within slice s at the end of round k.
$t_{i,s}(k)$	Airtime assigned to queue i within slice s in round k.
T_{max}	The maximum possible transmission time of a packet.
p_s	Ratio of airtime requested by a slice s.
$T_{i,s}(t_1, t2)$	The airtime assigned to queue i within slice s in the time interval (t_1, t_2).
$T_s(t_1, t_2)$	The airtime assigned to all queues of slice s in the interval (t_1, t_2).
$\mathcal{S}$	The set of all slices instantiated on an AP.
$N_{\mathcal{S}}$	Total number of slices on an AP.
$\mathcal{U}$	The set of users associated to an AP.
$N_{\mathcal{U}}$	The total number of user associated to an AP.
$\mathcal{U}_s$	The set of users that belong to slice s.
N_s	Number of users that belong to slice s.
q_s	Quantum of every queue of the slice s.
Q_s	The sum of all the quantums of slice s.
Q	The sum of all the quantums of the system.

5 Analytical Study

For this study we propose the following scenario: a slice tenant requests a share of the total airtime, as a quantified value between 0 and 1, accepting a tolerance for the possible variation of its request, also quantified by a value between 0 and 1. For example, if a tenant requests a share of 0.3 of the total airtime with a tolerance of 0.1, it means that we must guarantee an airtime share that varies between 0.27 and 0.33. This guarantee is feasible for a given allocation window.

The objectives of this study are: (i) to find a lower bound to the size of the allocation window for a given request and tolerance, (ii) to find a measure of the achievable fairness among users of a slice, and (iii) to study the latency introduced by the ATERR scheduling.

For the analysis we consider the set *active users* as the subset of users associated with the AP with pending incoming traffic. Hence, the queues assigned to the active users at the AP always have packets to be dequeued.

5.1 Allocation-window Size

Regarding objective (i), as a first step, in Lemma 1, we found the airtime a slice s gets in any time interval. Then, in Lemma 2, using this result we bound the ratio between a slice's airtime and the total allocated airtime. Finally, in Theorem 1, we propose a lower bound for the allocation-window's size.

Lemma 1 *In any interval of time (t_1, t_2) such that the set of active users do not vary and where ATERR can service each queue of the system m times, we have that the airtime assigned to any slice s within that interval satisfies:*

$$mQ_s - N_sT_{max} \leq T_s(t_1, t_2) \leq mQ_s + N_sT_{max} \quad \forall s \in \mathcal{S} \tag{5}$$

Proof As we already mentioned, ATERR dequeues packets from a queue i within the slice s until the excess of the queue i becomes non-negative. Then, we have that at the end of each round the excess satisfies that:

$$0 \leq e_{i,s} < T_{max} \tag{6}$$

where T_{max} is the maximum possible airtime of a packet. This value is given by the largest packet size and the lowest transmission rate possible in the system.

From the definition of the algorithm, we have that the airtime excess at a round k is the airtime excess from the previous round plus the time assigned in the current round minus the quantum.

$$e_{i,s}(k) = e_{i,s}(k-1) + t_{i,s}(k) - q_s \tag{7}$$

where $t_{i,s}(k)$ is the time assigned to queue i from slice s in the round k.

Then, rearranging terms we have that the time assigned at round k is:

$$t_{i,s}(k) = q_s + e_{i,s}(k) - e_{i,s}(k-1) \tag{8}$$

From its definition we also have that the total airtime assigned to queue i from slice s in the time interval (t_1, t_2) is:

$$T_{i,s}(t_1, t_2) = \sum_{k=1}^{m} t_{i,s}(k) \tag{9}$$

Replacing (8) in (9):

$$T_{i,s}(t_1, t_2) = \sum_{k=1}^{m} (q_s + e_{i,s}(k) - e_{i,j}(k-1)) \tag{10}$$

$$T_{i,s}(t_1, t_2) = mq_s + \sum_{k=1}^{m} (e_{i,s}(k) - e_{i,s}(k-1)) \tag{11}$$

$$T_{i,s}(t_1, t_2) = mq_s + e_{i,s}(m) - e_{i,s}(0) \tag{12}$$

Finally, from (6) and (12) we have:

$$mq_s - T_{max} \leq T_{i,s}(t_1, t_2) \leq mq_s + T_{max} \tag{13}$$

Therefore, the total time assigned to a flow in the time interval (t_1, t_2) can vary between those two values and the actual value on each interval will depend on the specific sequence of packets dequeued. Hence, the time assigned to a slice s in the interval (t_1, t_2), $T_s(t_1, t_2) = \sum_{i=1}^{N_s} T_{i,s}(t_1, t_2)$ is bound by:

$$m\sum_{i=1}^{N_s} q_s - N_s T_{max} \leq T_s(t_1, t_2) \leq m\sum_{i=1}^{N_s} q_s + N_s T_{max} \tag{14}$$

□

This result provides bounds on the amount of airtime the algorithm can assign to a slice in a given time interval. As expected, the allocated airtime is given by the amount of times the queues of the slices are served (m) multiplied by the airtime assigned on each service (Q_s). However, as the algorithm can assign in excess in some rounds and compensate with a deficit in following rounds, the actual assignment will depend in what happens in the first and last round. Hence, the result follows from considering the maximal possible airtime excess assigned to a slice in a round, which is given by the number of queues of the slice and the maximal possible airtime to be allocated.

Lemma 2 *In any interval of time (t_1, t_2) such that the active users set does not vary and where ATERR can service each queue of the system m times, the ratio of airtime r assigned to any slice s of the system satisfies:*

$$\frac{mQ_s - N_s T_{max}}{mQ + \hat{N} T_{max}} \leq r_s(t_1, t_2) \leq \frac{mQ_s + N_s T_{max}}{mQ - \hat{N} T_{max}} \quad \forall s \in \mathcal{S} \tag{15}$$

where $\hat{N} = N - 2N_s$.

Proof The airtime ratio allocated to a slice s in the interval (t_1, t_2) will be given by the time assigned to the slice s over the total time assigned to all slices:

$$r_s(t_1, t_2) = \frac{T_s(t_1, t_2)}{\sum_{l=1}^{S} T_l(t_1, t_2)} \tag{16}$$

Applying the bounds from Lemma 1 we obtain:

$$\frac{T_s(t_1, t_2)}{\sum\limits_{l=1}^{S} T_l(t_1, t_2)} \geq \frac{mQ_s - N_s T_{max}}{m\sum\limits_{l=1}^{S} Q_l + \sum\limits_{l=1, l\neq s}^{S} N_l T_{max} - N_s T_{max}} \tag{17}$$

$$\frac{T_s(t_1, t_2)}{\sum\limits_{l=1}^{S} T_l(t_1, t_2)} \leq \frac{mQ_s + N_s T_{max}}{m\sum\limits_{l=1}^{S} Q_l - \sum\limits_{l=1, l\neq s}^{S} N_l T_{max} + N_s T_{max}} \tag{18}$$

The lower bound consists on allocating the shortest possible time to the slice s and the longest time to all the other slices. The upper bound is achieved when allocating the longest time to slice s while allocating the shortest time to the rest of the slices.

Operating we have:

$$\frac{T_s(t_1,t_2)}{\sum_{l=1}^{S} T_l(t_1,t_2)} \geq \frac{mQ_s - N_s T_{max}}{mQ + (N - 2N_s)T_{max}} \tag{19}$$

$$\frac{T_s(t_1,t_2)}{\sum_{l=1}^{S} T_l(t_1,t_2)} \leq \frac{mQ_s + N_s T_{max}}{mQ - (N - 2N_s)T_{max}} \tag{20}$$

which demonstrates the lemma.

□

We have thus obtained bounds on how much the airtime allocation can vary from the requested ratio. This variation depends on some controllable parameters such as the number of active queues N, the number of active queues in the slice N_s, the sums of quantums Q_s and Q, and the number of evaluation rounds m within the time interval (t_1, t_2).

Hence, we can establish the required parameters to keep the variation bounded by some tolerance.

For the sake of simplicity, we assume the following:[1]

Assumption 1 *We assume that the AP is placed in a quiet transmission medium so that packets can be transmitted immediately when dequeued.*

Theorem 1 *ATERR guarantees the required airtime share p_s to any slice $s \in \mathcal{S}$ with a tolerance of $K \in (0,1]$ in a time window of size $W_{s,K}$, which satisfies:*

$$W_{s,K} \geq \frac{T_{max}}{Kp_s}\left(p_s\hat{N} + N_s + \sqrt{(p_s\hat{N} + N_s)^2 + (Kp_s\hat{N})^2}\right) - NT_{max} \tag{21}$$

Proof From Lemma 2 we have an upper and lower bound for the airtime ratio assigned to a slice. We want to confine the difference between the lower and upper bounds such that:

$$\frac{mQ_s + N_s T_{max}}{mQ - \hat{N}T_{max}} - \frac{mQ_s - N_s T_{max}}{mQ + \hat{N}T_{max}} \leq Kp_s \tag{22}$$

Operating, and using $Q_s = p_s Q$, we obtain:

$$\frac{2QT_{max}(p_s\hat{N} + N_s)m}{Q^2m^2 - \hat{N}^2T_{max}^2} \leq Kp_s \tag{23}$$

Rewriting the previous equation as a function of m we get:

$$-Kp_sQ^2m^2 + 2QT_{max}(p_s\hat{N} + N_s)m + Kp_s\hat{N}^2T_{max}^2 \leq 0 \tag{24}$$

The positive solution of (24) is:

$$m \geq \frac{T_{max}}{Kp_sQ}(p_s\hat{N} + N_s + \sqrt{(p_s\hat{N} + N_s)^2 + (Kp_s\hat{N})^2}) \tag{25}$$

[1] This assumption disregards the time that a packet may be waiting for the transmission medium to be clear, with an impact on the accuracy of our analysis. Including such time in our model is left for further work.

Hence, we have the minimum number of rounds that are needed to guarantee the allocation of airtime within some tolerance K. The time required for this m rounds depends on the number of queues in the system, and on the time each queue needs to transmit. It may also depend on some external variables, such as the transmission opportunities the device obtains from the medium. From Assumption 1, we have that the time consumed by the m rounds only depends on internal system parameters.

From previous results we know that the total time consumed in m rounds is:

$$T(m) = T(t_1, t_2) = \sum_{l=1}^{S} T_l(t_1, t_2) \geq mQ - NT_{max} \tag{26}$$

Then, the theorem follows from substituting the minimum m from (25) in (26).

□

Hence, Theorem 1 gives a lower bound for the allocation window needed to guarantee the required airtime share to any slice with an error lower than K. The obtained bound is a function of the number of associated users to the AP (N) and the number of users in the slice (N_s). This result is very important for the negotiation of slice requirements with a tenant, and to implement an access control mechanism for new users or slices to an AP. In what follows we illustrate this result with numerical values.

5.1.1 Graphical Illustration

In Figure 2 we plot numerical values for the lower bound of W given in Theorem 1 with a value of $K = 0.1$ and different airtime ratio requirements (p_s). We choose T_{max} as $10ms$, which is the maximum time defined in IEEE 802.11 standard [25] for an aggregate frame.

It is important to notice that these results are lower bounds, which appear in the worst case scenario. This is very unlikely to happen, as it implies getting the lowest assignment for all the queues of the considered slice and, at the same time, the highest airtime assignment for all the other slices of the system with the minimum possible data rate.

As we show in Section 6, in more common scenarios, the requested assignments are correctly achieved in smaller allocation windows.

5.2 Fairness Among users of one Slice

Regarding our objective (ii), following a similar analysis to the one proposed in [9] we can show that within a slice all queues receive a fair share of the airtime allocated to that slice.

Lemma 3 *In any interval of time (t_1, t_2) such that the active user set does not vary and where ATERR can service each queue of the system m times, the airtime allocated to a queue i within a slice s satisfies:*

$$mq_s - T_{max} \leq T_{i,s}(t_1, t_2) \leq mq_s + T_{max} \quad \forall i \in \mathcal{U}_s \;\; \forall s \in \mathcal{S} \tag{27}$$

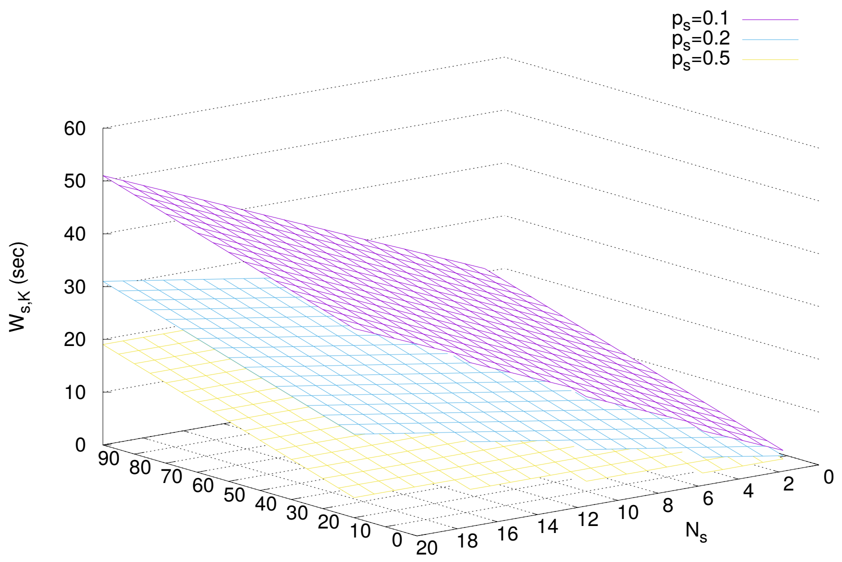

Fig. 2 Worst case lower bounds for $W_{j,K}$ with a tolerance of 0.1 as a function of the number of associated users N and number of users in the slice N_s

The proof is analogue to the one of Lemma 1.

The following theorem gives an upper bound on the difference of airtime allocation between any two queues of the same slice. As can be seen, this bound depends on the quantum size and T_{max}.

Theorem 2 *In any interval of time* (t_1, t_2) *such that the active user set does not vary, the difference of airtime assigned to any two queues* i *and* j *of a slice* s *satisfies:*

$$T_{i,s}(t_1, t_2) - T_{j,s}(t_1, t_2) \leq q_s + 2T_{max} \tag{28}$$

Proof For the proof we will take the worst case, which is given when $T_{i,s}(t_1, t_2)$ is allocated the longest possible airtime and $T_{j,s}(t_1, t_2)$ the shortest. In any interval (t_1, t_2) where the algorithm service queue i for m times, we know from Lemma 3 that:

$$T_{i,s}(t_1, t_2) \leq mq_s + T_{max} \tag{29}$$

In that same interval, the algorithm can serve queue l for m' times, then from Lemma 3 we have:

$$T_{j,s}(t_1, t_2) \geq m'q_s - T_{max} \tag{30}$$

Because of the *round robin* behaviour of ATERR it happens that $m - 1 \leq m' \leq m$, and therefore the smallest value of $T_{j,s}(t_1, t_2)$ happens when $m' = m - 1$. Hence, the difference is:

$$T_{i,s}(t_1, t_2) - T_{j,s}(t_1, t_2) \leq mq_s + T_{max} - m'q_s - T_{max} \tag{31}$$

$$T_{i,s}(t_1, t_2) - T_{j,s}(t_1, t_2) \leq mq_s - (m - 1)q_s + 2T_{max} \tag{32}$$

Finally we get:

$$T_{i,s}(t_1,t_2) - T_{j,s}(t_1,t_2) \leq mq_s - (m-1)q_s + 2T_{max} \tag{33}$$
$$T_{i,s}(t_1,t_2) - T_{j,s}(t_1,t_2) \leq q_s + 2T_{max} \tag{34}$$

□

5.3 Latency Analysis

Regarding our objective (iii), in this section we find a lower bound on the time between two consecutive service rounds of the ATERR algorithm. With this result we find the parameters that can affect the latency caused by the proposed scheduling algorithm. In particular, we find a lower bound, which depends on the number of queues and on the value of the *quantum*.

Theorem 3 *The time l_i between two consecutive service rounds to a queue i in a slice s, is bounded by:*

$$l_i < Q - q_s + (N-1)T_{max} \tag{35}$$

Proof Lets consider a packet P which was left at the head of the queue i after round R. Lets call l_i to the time the packet P waits at the queue until its transmission starts. P will have to wait while all other queues in the system are served. From equations (8) and (6) we know that a queue in slice s will be allocated, at most, an airtime of $q_s + T_{max}$, which happens when the previous excess is zero and a packet with T_{max} is dequeued[2]

Hence, the upper bound for the latency is reached when all other queues from all the slices are served with this maximum airtime.

$$l_i < \sum_{j=1}^{N_S} \sum_{\substack{k=1 \\ k \neq i}}^{N_j} q_k + T_{max} \tag{36}$$

$$l_i < \sum_{j=1}^{N_S} \sum_{k=1}^{N_j} q_k + T_{max} - (q_s + T_{max}) \tag{37}$$

$$l_i < \sum_{j=1}^{N_S} \sum_{k=1}^{N_j} q_k + (N-1)T_{max} - q_s \tag{38}$$

$$l_i < Q - q_s + (N-1)T_{max} \tag{39}$$

□

This result shows that despite making the *quantum* very small, the latency cannot be bounded, and it still depends on the number of queues. Even more, these two parameters (Q and N) are coupled, as with a fixed *quantum*, when N increases, Q also increases. Another outcome from Theorem 3 is that queues with a bigger *quantum* will have a lower latency.

In summary, the latency is given by:

[2] It is important to notice that this is valid only if at each round at least one packet can be dequeued. For this to happen it is required for the quantum to be at least equal to T_{max}.

– The number of queues in the system, which is given by the number of slices, and users per slice.
– The size of the *quantums*.
– The maximum packet airtime (T_{max}).

Regarding the total queue latency of a packet in an AP, it depends not only on the dequeuing latency but also on the arrival rate of packets to the queue. For the system to be stable and for the queue latency not to grow indefinitely we need the arrival rate to be lower than the serving time. A solution to make the latency not to grow indefinitely could be to use some *Queue Management* scheme on each queue, to discard packets and to take advantage of the TCP congestion control when possible.

6 Experimental Evaluation

As a proof of concept we have implemented ATERR in the NS3 Network Simulator [26], by modifying the existent transmission queuing structure of the wireless module. The simulation code is available at [27]. In this section, we depict simulation results showing how our analytical model correctly predicts the algorithm behaviour. Additionally, we show an evaluation for ATERR in different scenarios.

6.1 Evaluation of the Allocation-window Size

This experiment evaluates the airtime allocation achieved by ATERR while varying the allocation-window size. In particular, we measure the airtime-share allocated to each queue, depending on the size of the allocation window, the number of users in a slice and the airtime-allocation requests.

The experiment is configured as follows: there are 40 clients connected to one AP; the AP has two slices defined, being s the slice under analysis. Slice s has an airtime-allocation request of p_s, and the number of clients in slice s is N_s. We run experiments with different values for p_s and N_s, and measure the airtime-share allocated under different allocation-window sizes. The AP generates a Constant Bit Rate (CBR) UDP traffic to all the clients, with a high rate so as to saturate the channel. The clients are deployed around the AP and move randomly inside a square of $25m \times 25m$, so as to generate variability in the transmission rates and thus in the consumed airtime.

We run experiments for two different airtime requests (p_s): 0.1 and 0.2. Figures 3 and 4 show the variation of the allocated airtime-share for 0.1 and 0.2 requests. This variation is calculated as the amplitude between the maximum and minimum airtime-share allocated (normalized by p_s) through the experiment. The objective is to show that the variation satisfies the bounds found in Section 5.1.

There are four plots on each figure which depict the results obtained with different number of users in the analyzed slice: 5, 10, 15 and 20. The dotted horizontal line marks a desired error tolerance (K) of 0.1. The dotted vertical lines represent the analytical bounds for each N_s.

It can be observed that, as predicted by the theoretical analysis, the airtime allocation provided by the mechanism can not be guaranteed under any allocation-window size. It is also possible to observe that in the simulation the requested

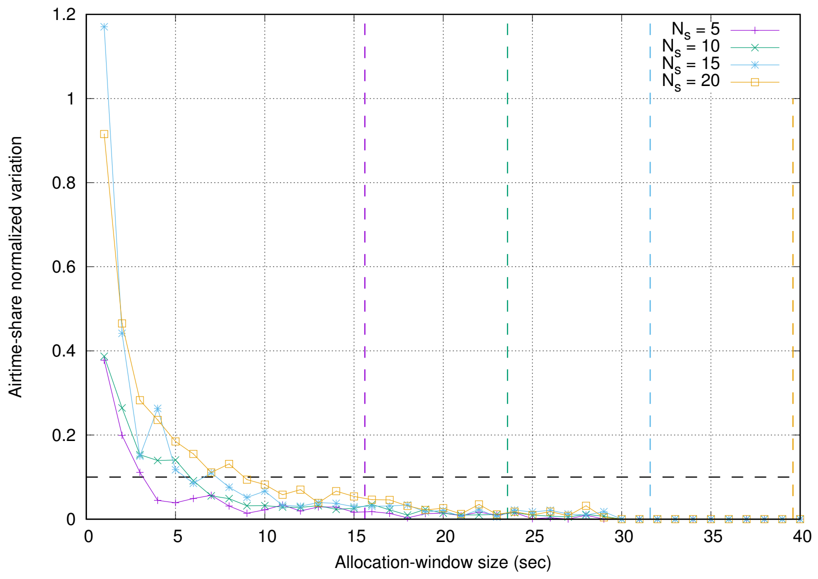

Fig. 3 Allocated airtime-share variation vs. allocation-window size for a requested airtime-share of 0.1.

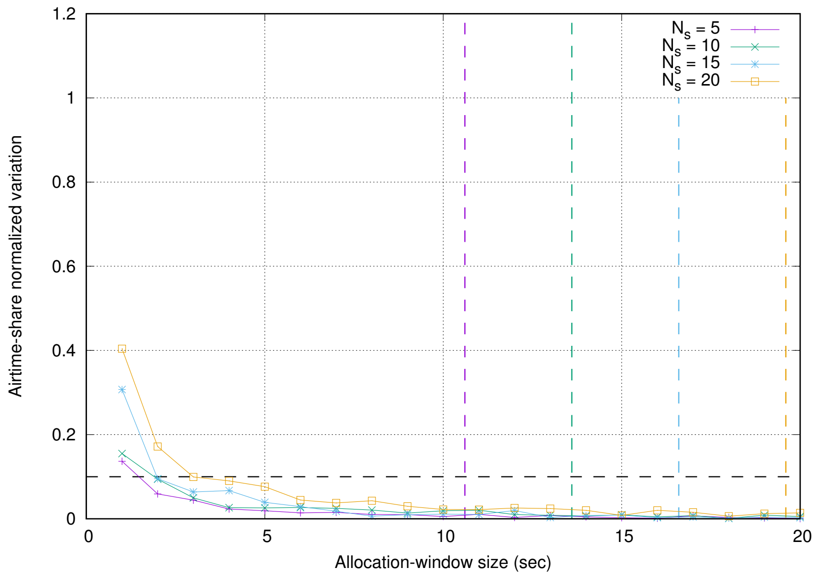

Fig. 4 Allocated airtime-share variation vs. allocation-window size for a requested airtime-share of 0.2.

allocation and tolerance is achieved at smaller values for the allocation-window size than the lower theoretical bound. For example, for the case of $p_s = 0.1$ and $N_s = 20$, the tolerance is achieved with a window of 10 s, while the bound is 28.4 s.

Furthermore, we can see the impact of the two variables evaluated in the experiment: the number of users in the slice and the airtime-share requested. It can be seen that as the number of users in the slice increases, it becomes more difficult to achieve the desired variation tolerance. The opposite happens with the requested airtime-share, as the requested airtime-share becomes smaller the needed allocation-window for achieving the desired tolerance increases.

In conclusion, these results corroborate the previous analysis. As mentioned earlier, it is important to notice that the calculated bounds are considering the worst possible case, which explains the differences with the experimental results.

6.2 Evaluation with UDP Traffic

The experiment consists on a deployment comprising one AP and ten users randomly located around the AP. Three slices are defined:

– Slice 1 requests 20% of the airtime ($p_s = 0.2$) and there are four users associated with the slice (U1, U2, U3 and U4).
– Slice 2 requests 20% of the airtime ($p_s = 0.2$) and there are four users associated with the slice (U4, U5, U6 and U7).
– Slice 3 requests 60% of the airtime ($p_s = 0.6$) and there are four users associated with the slice (U7, U8, U9 and U10).

It is worth noticing that U4 and U7 belong to two slices.

In this experiment we evaluate two scenarios, one with static users and one with mobile users. For the static case, users are randomly deployed around the AP with an average distance from the AP of 5 m. For the other case, users move inside a square of 50 $m \times 50$ m with the AP at the center, following a *Random Direction* mobility model with a random speed between 1 and 2 m/s. The AP is configured to transmit Constant Bit Rate (CBR) UDP traffic to every user with a high rate so as to maintain all queues backlogged. Simulations are repeated 20 times, varying the users' location and each simulation lasts 60 seconds for the static scenario and 120 seconds for the mobile one. We measured the airtime allocated to each slice with an allocation-window of 1 second.

Figures 5 and 6 show an histogram of the airtime-shares assigned to each slice in all the experiments executed for the static and mobile case respectively. As can be seen, the requested proportion of airtime is correctly allocated to each slice and with variability inside the tolerance of 0.1 from the requested airtime-share.

It can also be seen from the figures that the experiments show more variability in the airtime-share allocation for the mobile scenario. This can be explained due to the changes in the channel conditions caused by the varying distance between the AP and the clients. These changes in the channel condition impact on the transmission rate and thus on the airtime consumed by the different clients.

We also evaluated the airtime-share assigned to each user on every slice to assess the fairness among users of the same slice. The results are shown in Figure 7. For the sake of clarity the figure shows the result for one simulation execution. The figure highlights that the assignments for each user are noticeably fair. To better assess the fairness, we calculated the Jain's fairness index of the assigned airtime to each user within the slices for 20 independent simulations. The results show that for all slices, the index is greater than 0.999, which is indeed a very good result.

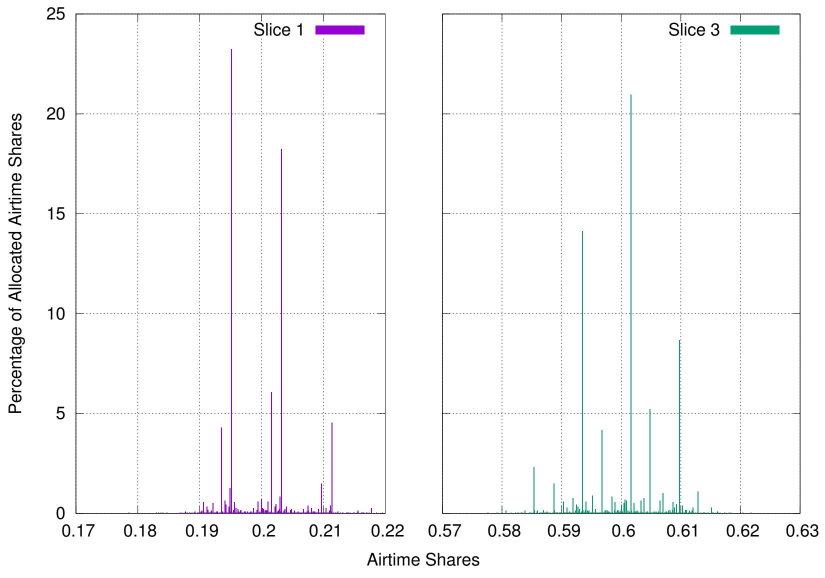

Fig. 5 Histogram of the airtime-shares allocated in the static scenario for Slice 1 ($p_s = 0.2$) and Slice 3 ($p_s = 0.6$).

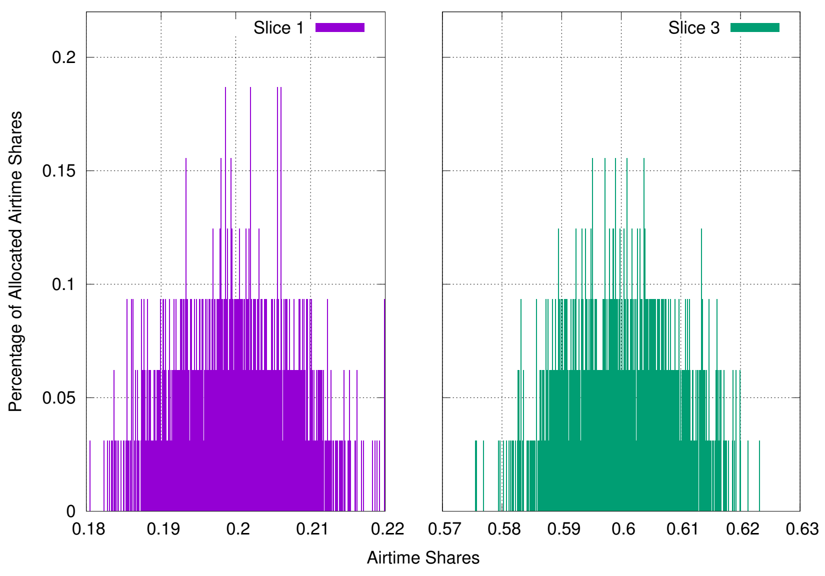

Fig. 6 Histogram of the airtime-shares allocated in the mobile scenario for Slice 1 ($p_s = 0.2$) and Slice 3 ($p_s = 0.6$).

6.3 Evaluation with TCP Traffic

In this experiment we maintain the same configuration as the previous one, but we use a constant flow of TCP traffic. The traffic generator consists on an application

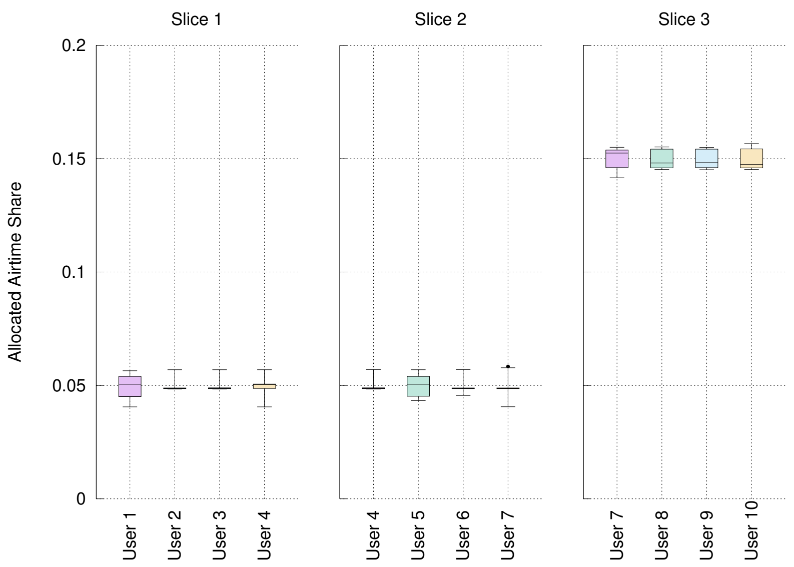

Fig. 7 Allocated airtime-share to each user of the slices.

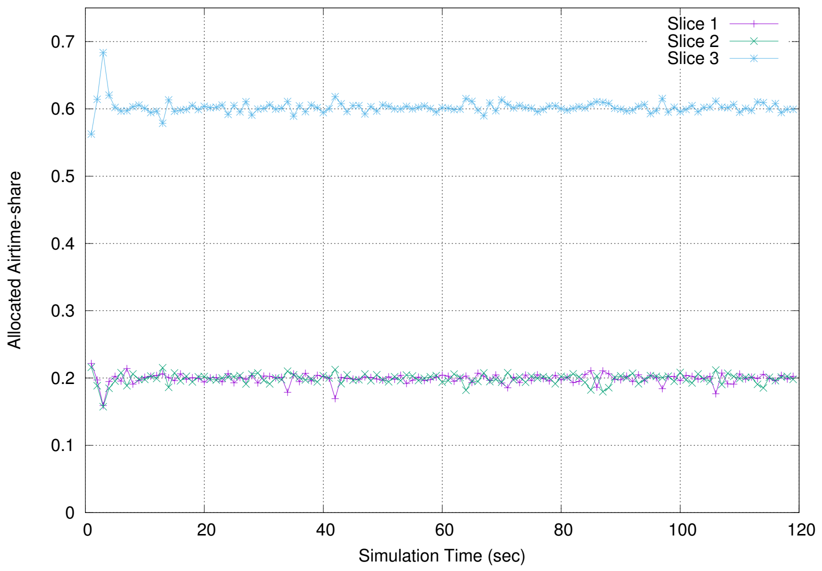

Fig. 8 Airtime-share allocation for TCP traffic.

which sends data as fast as possible, as it would be the case of a 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. TCP brings more challenges for the algorithm as the congestion and flow control will generate variable traffic load making the queue sizes to fluctuate.

Figure 8 shows the airtime assignment for the three slices during the entire simulation time. We observe that at the beginning the allocation minimally fluctuates, and after 5 seconds it stabilizes on the requested ratios. We can observe small fluctuations during the experiments. These are due to the TCP congestion control behaviour, which causes the queues to empty, leading to allocate the unused resources to other queues.

Figure 9 depicts an histogram of the airtime-shares assigned to each slice in all the executed experiments. We maintain an allocation-window size of 1 second so as to compare with the UDP experiment. It can be observed that most of the airtime-share allocations are within the requested values but, differently from UDP traffic, the variation is larger. However, with an allocation-window of 4 seconds, the variation is within the expected tolerance of 0.1, as can be seen in Figure 10.

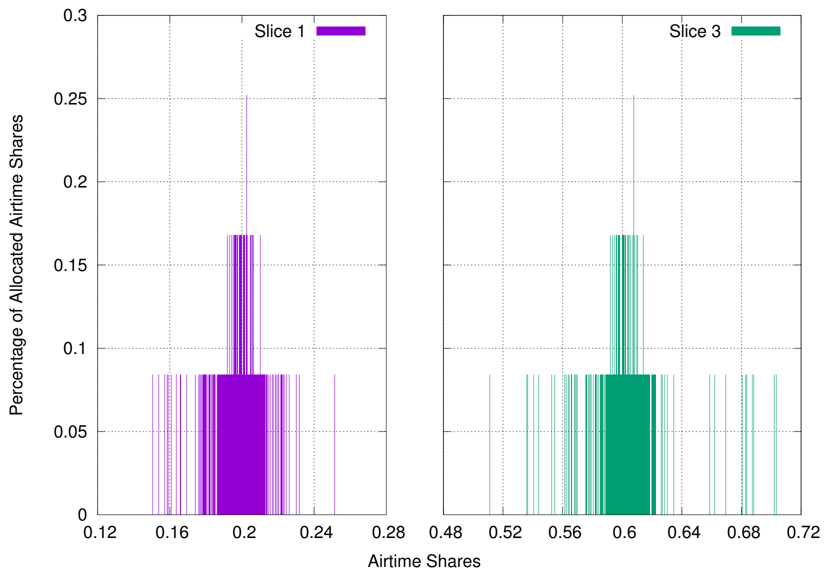

Fig. 9 Histogram of airtime-shares allocated in the TCP scenario for Slice 1 ($p_s = 0.2$) and Slice 3 ($p_s = 0.6$). Allocation-window of 1s.

6.4 Evaluation with UDP Fluctuating Traffic

This experiment is very similar to the previous one, but in this case, we use variable traffic loads. The objective is to show how the resources are automatically reallocated when some of the slices do not use all the requested resources.

The traffic configuration is as follows:

– At time 0, all slices use all the requested resources.
– At 30 seconds, traffic to all users of slice 3 is halved.
– At 60 seconds, traffic to all users of Slice 1 is stopped.
– At 90 seconds, all traffic is restored.

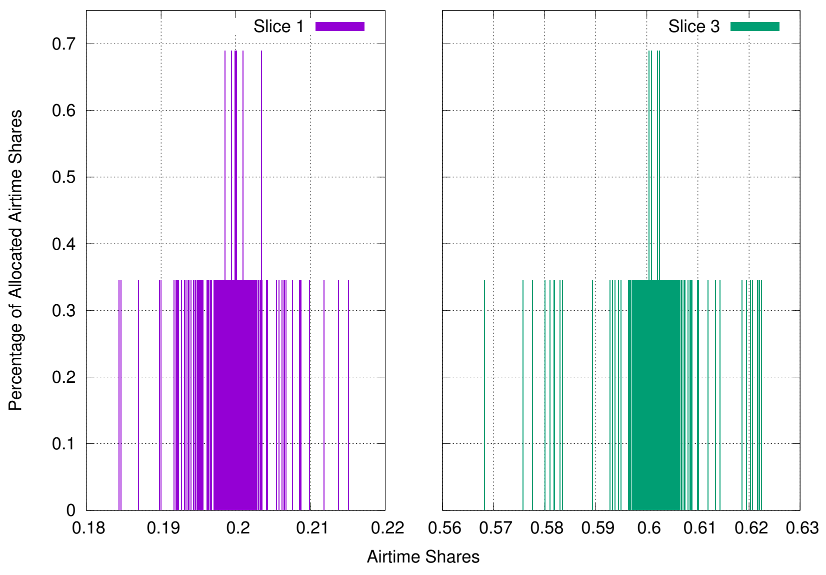

Fig. 10 Histogram of airtime-shares allocated in the TCP scenario for Slice 1 ($p_s = 0.2$) and Slice 3 ($p_s = 0.6$). Allocation-window of 4s.

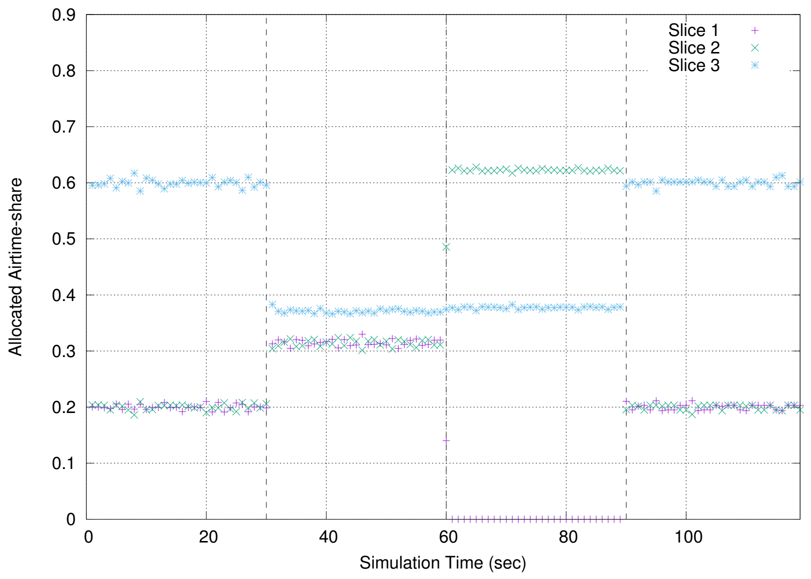

Fig. 11 Allocated airtime-share vs. simulation time for fluctuating UDP traffic.

Figure 11 shows the results of the experiment. We can highlight the following observations: during the first 30s the algorithm allocates the resources exactly as requested, because each slice is using all the assigned resources. In the interval from second 30 to 60, Slice 3 reduces its traffic, and does not use all the assigned resources which results in Slice 1 and 2 using the remaining resources (20%) fairly. In the interval from second 60 to 90, Slice 1 does not use any of its resources, which allows Slice 2 to use all the available resources; finally, at second 90 all the traffic is restored, and the assignment to each slice is recovered.

Previous results also illustrate the isolation achieved by ATERR, as the resource assignments are not modified because of traffic changes on other slices. Even more, these results show that the algorithm adjusts to the new traffic patterns almost instantly.

7 Conclusions

This work proposes a resource allocation mechanism based on airtime assignment to achieve infrastructure sharing and slicing in WiFi Access Points. The approach is aimed to be simple and has the potential to be easily used in different infrastructure sharing scenarios.

The novelty of the approach is in the use of airtime allocation through packet scheduling for wireless slicing. This strategy has many advantages, such as being adaptive to traffic load and avoiding the modification of MAC-layer parameters.

The work also contributes with a theoretical analysis of the mechanism, showing the parameters that influence the airtime allocation, and providing the necessary tools to guarantee a requested airtime-share. Thanks to the proposed model it is possible to determine if an airtime-share request can be accepted. This would help managing the slices, allowing the implementation of user access-control. It is also an important tool to guarantee that the slice implementation complies with the allocation agreement between the tenant and the slicing provider.

Furthermore, through extensive simulation experiments we have shown the correct operation of the proposed mechanism. In particular, we have studied the key characteristics of the proposal under different scenarios. The experiment results specifically illustrate: the accurate airtime-share allocation based on slice requests, an acceptable airtime fairness among users of a slice, and the efficient utilization and isolation of resources between slices.

We are currently working towards the extension of this mechanism to provide throughput and/or delay guarantees to also support *QoS Slicing*. Moreover, we also plan to deploy the proposed scheme over real platforms, both to assess its feasibility and to broaden its performance analysis.

Acknowledgements 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. Li, Q., Niu, H., Wu, G., Fong, M.H., Papathanassiou, A.: Slicing architecture for wireless communication (2017). US Patent 9,775,045
2. Kusume, K., Fallgren, M., Queseth, O., Braun, V., Gozalvez-Serrano, D., Korthals, I., Zimmermann, G., Schubert, M., Hossain, M.I., Widaa, A.A., Chatzikokolakis, K., Holakouei, R., Jeux, S., Hernando, J.L., Boldi, M.: Deliverable D1.5. Updated scenarios, requirements

and KPIs for 5G mobile and wireless system with recommendations for future investigations. (2015). URL `https://www.metis2020.com/wp-content/uploads/deliverables/METIS_D1.5_v1.pdf`

3. Ericsson: 5G systems (white paper). Tech. Rep. Uen 284 23-3244, Ericsson (2015). URL `http://www.ericsson.com/res/docs/whitepapers/what-is-a-5g-system.pdf`
4. Richart, M., Baliosian, J., Serrat, J., Gorricho, J.L., Agüero, R., Agoulmine, N.: Resource allocation for network slicing in wifi access points. In: IEEE (ed.) Network and Service Management (CNSM), 2017 13th International Conference on. IEEE (2017)
5. Heusse, M., Rousseau, F., Berger-Sabbatel, G., Duda, A.: Performance anomaly of 802.11 b. In: INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, vol. 2, pp. 836–843. IEEE (2003)
6. Høiland-Jørgensen, T., Kazior, M., Täht, D., Hurtig, P., Brunstrom, A.: Ending the anomaly: Achieving low latency and airtime fairness in wifi. In: 2017 USENIX Annual Technical Conference (USENIX ATC 17), pp. 139–151. USENIX Association, Santa Clara, CA (2017). URL `https://www.usenix.org/conference/atc17/technical-sessions/presentation/hoilan-jorgesen`
7. Riggio, R., Miorandi, D., Chlamtac, I.: Airtime deficit round robin (adrr) packet scheduling algorithm. In: Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on, pp. 647–652. IEEE (2008)
8. Garroppo, R.G., Giordano, S., Lucetti, S., Tavanti, L.: Providing air-time usage fairness in ieee 802.11 networks with the deficit transmission time (dtt) scheduler. Wireless Networks **13**(4), 481–495 (2007)
9. Shreedhar, M., Varghese, G.: Efficient fair queueing using deficit round robin. SIGCOMM Comput. Commun. Rev. **25**(4), 231–242 (1995). DOI 10.1145/217391.217453. URL `http://doi.acm.org/10.1145/217391.217453`
10. Aboba, B.: Virtual access points. Tech. Rep. IEEE 802.11-03/154r0, Microsoft, One Microsoft Way, Redmond, WA 98052-6399 (2003)
11. Xia, L., Kumar, S., Yang, X., Gopalakrishnan, P., Liu, Y., Schoenberg, S., Guo, X.: Virtual wifi: Bring virtualization from wired to wireless. In: Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, VEE '11, pp. 181–192. ACM, New York, NY, USA (2011). DOI 10.1145/1952682.1952706. URL `http://doi.acm.org/10.1145/1952682.1952706`
12. Aljabari, G., Eren, E.: Virtualization of wireless lan infrastructures. In: Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2011 IEEE 6th International Conference on, vol. 2, pp. 837–841. IEEE (2011)
13. Richart, M., Baliosian, J., Serrat, J., Gorricho, J.L.: Resource slicing in virtual wireless networks: A survey. IEEE Transactions on Network and Service Management **13**(3), 462–476 (2016)
14. Banchs, A., Serrano, P., Patras, P., Natkaniec, M.: Providing throughput and fairness guarantees in virtualized wlans through control theory. Mob. Netw. Appl. **17**(4), 435–446 (2012). DOI 10.1007/s11036-012-0382-2. URL `http://dx.doi.org/10.1007/s11036-012-0382-2`
15. Nakauchi, K., Shoji, Y., Nishinaga, N.: Airtime-based resource control in wireless lans for wireless network virtualization. In: Ubiquitous and Future Networks (ICUFN), 2012 Fourth International Conference on, pp. 166–169. IEEE (2012)
16. Guo, K., Sanadhya, S., Woo, T.: Vifi: virtualizing wlan using commodity hardware. ACM SIGMOBILE Mobile Computing and Communications Review **18**(3), 41–48 (2015)
17. Derakhshani, M., Wang, X., Le-Ngoc, T., Leon-Garcia, A.: Virtualization of multi-cell 802.11 networks: Association and airtime control. arXiv preprint arXiv:1508.03554 (2015)
18. Katsalis, K., Choumas, K., Korakis, T., Tassiulas, L.: Virtual 802.11 wireless networks with guaranteed throughout sharing. In: Computers and Communication (ISCC), 2015 IEEE Symposium on, pp. 845–850. IEEE (2015)
19. Bhanage, G., Vete, D., Seskar, I., Raychaudhuri, D.: Splitap: leveraging wireless network virtualization for flexible sharing of wlans. In: Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE, pp. 1–6. IEEE (2010)
20. Bhanage, G., Daya, R., Seskar, I., Raychaudhuri, D.: Vnts: A virtual network traffic shaper for air time fairness in 802.16 e systems. In: Communications (ICC), 2010 IEEE International Conference on, pp. 1–6. IEEE (2010)
21. van de Belt, J., Ahmadi, H., Doyle, L.E.: Defining and surveying wireless link virtualization and wireless network virtualization. IEEE Communications Surveys & Tutorials **19**(3), 1603–1627 (2017)

22. Zaki, Y., Zhao, L., Goerg, C., Timm-Giel, A.: Lte wireless virtualization and spectrum management. In: Wireless and Mobile Networking Conference (WMNC), 2010 Third Joint IFIP, pp. 1–6. IEEE (2010)
23. Adiseshu, H., Parulkar, G., Varghese, G.: A reliable and scalable striping protocol. ACM SIGCOMM Computer Communication Review **26**(4), 131–141 (1996)
24. Hoeiland-Joergensen, T., McKenney, P., Taht, D., Gettys, J., Dumazet, E.: The flowqueue-codel packet scheduler and active queue management algorithm. Internet-Draft draft-ietf-aqm-fq-codel-06, IETF Secretariat (2016). URL `http://www.ietf.org/internet-drafts/draft-ietf-aqm-fq-codel-06.txt`. `http://www.ietf.org/internet-drafts/draft-ietf-aqm-fq-codel-06.txt`
25. IEEE Std 802.11^{TM}-2012, IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: WLAN MAC and PHY specifications. Online (2012)
26. NS-3 Network Simulator. URL `http://www.nsnam.org/`
27. Richart, M.: mrichart/ns-3-dev-git: Airtime slicing simulation code for ns3 (2018). DOI 10.5281/zenodo.1309919

Matías Richart is a PhD student at University of the Republic (Uruguay) and at Polytechnic University of Catalonia (Spain). He received his Computer Engineer and Master degree from the University of the Republic in 2011 and 2014 respectively. His research interests include autonomic control and resource management of wireless networks, virtual wireless networks, network slicing and network simulation.

Javier Baliosian received the degree of Computer Engineer from the University of the Republic (Uruguay) in 1998, and his PhD from the 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 in Swedenand and the Ericsson Ireland Research Centre where he worked as a researcher and project coordinator until late 2007. Currently, Javier is with the Department of Computer Science at the University of the Republic (Uruguay).

Joan Serrat received a degree of Telecommunications Engineering in 1977 and a PhD 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 special interest on 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 in University of Cantabria. He received his MSc in Telecommunications Engineering (1st class honors) in 2001 and the PhD (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 on Network Coding. He serves in the Editorial Board of IEEE Communication Letters and Wireless Networks (Springer). He is a senior member of IEEE since 2015.